



World Wide Web

# Web Services

## Proof-of-Concept

---

**Final Report of the COTS Web Services Workgroup**  
**September 2002**

Prepared for:  
**The Council on Technology Services**  
Commonwealth of Virginia

By:  
**The COTS Web Services Workgroup**  
Tim Bass, Chairman

September 13, 2002

Dear Colleagues:

In February 2002, the Council on Technology Services (COTS) initiated a work group to investigate the viability of Web Services technology as an application development, interoperability and integration approach. With this mission, in March 2002 a team was assembled with representatives from state agencies, institutions of higher education, localities and business partners that had serious interest in this new technology. All participants understood that this effort would require a significant commitment in terms of volunteering their time and technical resources, including various hardware platforms and software components. I am proud to say that over the past seven months, their dedication to the mission and willingness to extend all the resources necessary to get this job done have enabled us to successfully craft this report for your review and consideration.

Looking back over the past year, it is interesting to note how quickly the topic of Web Services has risen in terms of industry attention. Since COVITS 2001, Web Services has evolved from a basic discussion of proposed specifications and early-release software to a growing set of W3C and OASIS-approved web standards and production-grade software. While Web Services technology is by no means mature or problem-free, it seems clear that the marketplace sees its value in terms of interoperability and integration and is putting significant resources toward future enhancements.

On behalf of the workgroup, I would like to thank the Secretary of Technology, the COTS Executive Council, and the general membership for the opportunity to assess this new technology. It has truly been a valuable learning experience that will benefit the Commonwealth of Virginia in the months to come. On a personal level, I would like to thank all the workgroup participants for their strong commitment to the mission and schedule, the Department of Technology Planning (DTP) and the Department of Information Technology (DIT) for their extra support during the documentation phase, and the business partners for their willingness and ability to create and maintain a cooperative environment where typically competitive entities could work towards a common goal. I just can't say "thank you" enough.

Sincerely,

J. Timothy Bass  
Chief Technology Officer, Virginia Retirement System  
Chairman, COTS Web Services Workgroup

## Acknowledgements

The Web Services Workgroup consisted of numerous state agency, local agency, higher education, and private sector “volunteers.” We wish to extend our gratitude to our employers for their support in allowing us to participate in this dynamic Proof-of-Concept.

The workgroup would also like to thank the Secretary of Technology and the Council on Technology Services (COTS) members for their sponsorship and support on shaping this project.

As the workgroup designed the Proof-of-Concept project, many individuals participated in lively meetings at the Virginia Retirement System (VRS). The significant contribution of VRS is sincerely appreciated.

The reader will note that the workgroup teams followed an abbreviated planning methodology with required status reports. Many thanks for Nelly Romero of the Department of Medical Assistance Services for developing the methodology and for ensuring that each team submitted its report on time.

In addition to the workgroup members, many people took the time to read the draft of this document and provide useful recommendations that enriched our work. Harry Smith of the Virginia Department of Education corrected our grammar and punctuation. Janice Akers of the Department of Information Technology and Eric Perkins of the Department of Technology Planning took charge of combining multiple documents and writing styles into a final draft that made sense.

Finally, we wish to congratulate one another. Each member made a remarkable commitment to the project. The energy of the group was evident in every meeting, instant message, e-mail, and conference call. The energy put forth by every workgroup member is evidenced by the quality of the finished product you are about to read.

Members of the Web Services Workgroup

September 13, 2002

## Participants in the Web Service Proof-of-Concept Project

### Web Services Workgroup Chairman

Name	Organization	E-mail	Phone
Bass, Tim (Chair)	Virginia Retirement System	<a href="mailto:tbass@vrs.state.va.us">tbass@vrs.state.va.us</a>	804.344.3175

### Team 1 Members: DMV, Microsoft Corporation, Susquehanna Technologies

Name	Organization	E-mail	Phone
Strydom, William	Department of Motor Vehicles	<a href="mailto:dmvw1s@dmv.state.va.us">dmvw1s@dmv.state.va.us</a>	804.367.8402
Shelley, Lana	Department of Motor Vehicles	<a href="mailto:dmvlss@dmv.state.va.us">dmvlss@dmv.state.va.us</a>	804.367.2635
Mauney, Tim	Department of Motor Vehicles	<a href="mailto:dmvt1m@dmv.state.va.us">dmvt1m@dmv.state.va.us</a>	804.367.1379
Froggatt, David	Department of Motor Vehicles	<a href="mailto:dmvdcf@dmv.state.va.us">dmvdcf@dmv.state.va.us</a>	804.519.1069
Dodson, Debbie	Department of Information Technology	<a href="mailto:ddodson@dit.state.va.us">ddodson@dit.state.va.us</a>	804.786.5549
Kendrick, Don	Department of Information Technology	<a href="mailto:dkendrick@dit.state.va.us">dkendrick@dit.state.va.us</a>	804.371.5715
Vencil, Bill	Susquehanna Technologies	<a href="mailto:billy@susqtech.com">billy@susqtech.com</a>	540.665.3427
Hickman, Glenn	Susquehanna Technologies	<a href="mailto:glennh@susqtech.com">glennh@susqtech.com</a>	540.723.8700
DeHaven, Beth	Microsoft	<a href="mailto:bdehaven@microsoft.com">bdehaven@microsoft.com</a>	804.560.1118
Farlow, Hank	Microsoft	<a href="mailto:hankf@microsoft.com">hankf@microsoft.com</a>	804.560.7076
O'Keefe, Chuck	Microsoft	<a href="mailto:chuckok@microsoft.com">chuckok@microsoft.com</a>	540.247.8063
Xi, William	Microsoft	<a href="mailto:williamx@microsoft.com">williamx@microsoft.com</a>	602.315.0937
Fowler, Scott	VIPNet	<a href="mailto:sfowler@vipnet.org">sfowler@vipnet.org</a>	804.786.6220

### Team 2 Members: ABC, BEA Systems, Inc.

Name	Organization	E-mail	Phone
Hassen, David	Virginia Alcohol Beverage Control	<a href="mailto:dwhassn@abc.state.va.us">dwhassn@abc.state.va.us</a>	804.213.4492
Shirbacheh, Sue	Virginia Alcohol Beverage Control	<a href="mailto:sashirh@abc.sate.va.us">sashirh@abc.sate.va.us</a>	804.213.4725
Saneda, Chris	Virginia Alcohol Beverage Control	<a href="mailto:csaneda@abc.state.va.us">csaneda@abc.state.va.us</a>	840.213.4483
Howery, Will	BEA Systems	<a href="mailto:whowery@bea.com">whowery@bea.com</a>	804.921.7433
Lender, Dan	BEA Systems	<a href="mailto:daniel.lender@bea.com">daniel.lender@bea.com</a>	571.382.2454

Team 3 Members: VRS, DHRM, SilverStream, Mitem

Name	Organization	E-mail	Phone
Korb, Sonja	Virginia Retirement System	<a href="mailto:skorb@vrs.state.va.us">skorb@vrs.state.va.us</a>	804.649.8059 X 353
Oliver, John	Virginia Retirement System	<a href="mailto:joliver@vrs.state.va.us">joliver@vrs.state.va.us</a>	804.649.8059 X 691
Paul, Bradley	Department of Human Resource Mgt	<a href="mailto:bpaul@dhrm.state.va.us">bpaul@dhrm.state.va.us</a>	804.786.5877
Mira, Belchoir	Department of Human Resource Mgt	<a href="mailto:bmira@dhrm.state.va.us">bmira@dhrm.state.va.us</a>	804.225.2203
Kass, Allen	Department of Human Resource Mgt	<a href="mailto:akass@dhrm.state.va.us">akass@dhrm.state.va.us</a>	804.786.1932
Holcombe, Chris	SilverStream	<a href="mailto:cholcombe@silverstream.com">cholcombe@silverstream.com</a>	336.232.5620
O'Brien, Tanzeena	SilverStream	<a href="mailto:tobrien@silverstream.com">tobrien@silverstream.com</a>	703.245.6694
Cain, Barbara	SilverStream	<a href="mailto:bcain@silverstream.com">bcain@silverstream.com</a>	703.245.6695
Romiti, Mark	SilverStream	<a href="mailto:mromiti@silverstream.com">mromiti@silverstream.com</a>	703.624.1814
Pendergrass, Dave	MITEM	<a href="mailto:davep@mitem.com">davep@mitem.com</a>	703.532.2704

Team 4 Members: City of Virginia Beach, County of Roanoke, Sun Microsystems, T4 Consulting Group (T4CG)

Name	Organization	E-mail	Phone
Lam, Andrew	City of Virginia Beach	<a href="mailto:alam@vb.gov">alam@vb.gov</a>	757.430.6397
Pilcher, Joanne	City of Virginia Beach	<a href="mailto:jpilcher@vb.gov">jpilcher@vb.gov</a>	757.430.6371
Jamison, Andrea	City of Virginia Beach	<a href="mailto:ajamison@vb.gov">ajamison@vb.gov</a>	757.427.8160
Cowart, Gwen	City of Virginia Beach	<a href="mailto:gcowart@vb.gov">gcowart@vb.gov</a>	757.427.8160
Richardson, Beth	City of Virginia Beach	<a href="mailto:bmrichar@vb.gov">bmrichar@vb.gov</a>	757.427.8160
Bird, Nicole	Roanoke County	<a href="mailto:nbird@co.roanoke.va.us">nbird@co.roanoke.va.us</a>	540.561.8000 X 244
Ghee, Elton	Roanoke County	<a href="mailto:eghee@co.roanoke.va.us">eghee@co.roanoke.va.us</a>	540.561.8001
Cox, Wayne	Sun Microsystems	<a href="mailto:wayne.cox@sun.com">wayne.cox@sun.com</a>	703.204.4937
Wilkinson, David	Sun Microsystems	<a href="mailto:david.wilkinson@sun.com">david.wilkinson@sun.com</a>	919.816.4594
Martin, Jake	Sun Microsystems	<a href="mailto:jake.martin@sun.com">jake.martin@sun.com</a>	410.312.1750
Reed, Eric	Sun Microsystems	<a href="mailto:eric.reed@sun.com">eric.reed@sun.com</a>	703.280.3439
Polenur, Alexi	Sun Microsystems	<a href="mailto:Alexi.polenur@sun.com">Alexi.polenur@sun.com</a>	510.986.3814
Kasturi, Sekharam	Sun Microsystems	<a href="mailto:Sekharam.kasturi@sun.com">Sekharam.kasturi@sun.com</a>	408.404.8108
Hoehne, Paul	T4 Consulting Group	<a href="mailto:phoehne@t4cg.com">phoehne@t4cg.com</a>	703.729.4990
Field, Brian	T4 Consulting Group	<a href="mailto:bfield@t4cg.com">bfield@t4cg.com</a>	703.729.4990

Name	Organization	E-mail	Phone
Lytle, Bob	T4 Consulting Group	<a href="mailto:blytle@t4cg.com">blytle@t4cg.com</a>	703.729.4990

Team 5 Members: Virginia Department of Education, Virginia Department of Medical Assistance Services, Software AG, Inc.

Name	Organization	E-mail	Phone
Canada, Bethann	Department of Education	<a href="mailto:bcanada@mail.vak12ed.edu">bcanada@mail.vak12ed.edu</a>	804.225.2951
Hanzlik, David	Department of Education	<a href="mailto:dhanzlik@mail.vak12ed.edu">dhanzlik@mail.vak12ed.edu</a>	804.225.2946
Romero, Nelly	Department of Medical Assistance Services	<a href="mailto:nromero@dmas.state.va.us">nromero@dmas.state.va.us</a>	804.786.7353
Witz, Henry	Department of Medical Assistance Services	<a href="mailto:hwitz@dmas.state.va.us">hwitz@dmas.state.va.us</a>	804.225.3617
Clark, Sari	Software AG, Inc.	<a href="mailto:sari.clark@softwareagusa.com">sari.clark@softwareagusa.com</a>	703.391.8246
Taylor, John	Software AG, Inc.	<a href="mailto:John.taylor@softwareagusa.com">John.taylor@softwareagusa.com</a>	703.391.8119
Wood, Eric	Software AG, Inc.	<a href="mailto:eric.wood@softwareagusa.com">eric.wood@softwareagusa.com</a>	703.758.6832
Patterson, Joel	Software AG, Inc.	<a href="mailto:Joel.patterson@softwareagusa.com">Joel.patterson@softwareagusa.com</a>	703.391.8245
Ford, Trevor	Software AG, Inc.	<a href="mailto:Trevor.ford@softwareagusa.com">Trevor.ford@softwareagusa.com</a>	925.242.3752

Team 6 Members: George Mason University, WebMethods

Name	Organization	E-mail	Phone
Enache, Mihaela	George Mason University	<a href="mailto:menache@gmu.edu">menache@gmu.edu</a>	703.993.3566
Creuziger, John	George Mason University	<a href="mailto:jcreuzig@gmu.edu">jcreuzig@gmu.edu</a>	703.993.4160
Alefantis, LJ	WebMethods	<a href="mailto:LJ@webmethods.com">LJ@webmethods.com</a>	703.395.8333
Demory, Chris	WebMethods	<a href="mailto:cdemory@webMethods.com">cdemory@webMethods.com</a>	703.460.6067
Jones, Bob	WebMethods	<a href="mailto:Bob.jones@webmethods.com">Bob.jones@webmethods.com</a>	703.251.7155
Moore, Jaime	WebMethods	<a href="mailto:Jaime.Moore@webmethods.com">Jaime.Moore@webmethods.com</a>	703.460.5845
West, Floyd	WebMethods	<a href="mailto:fwest@webmethods.com">fwest@webmethods.com</a>	703.460.5935

UDDI Team Members: BEA, VIPNet

Name	Organization	E-mail	Phone
Howery, Will	BEA Systems	<a href="mailto:whowery@bea.com">whowery@bea.com</a>	804.921.7433
Fowler, Scott	VIPNet	<a href="mailto:sfowler@vipnet.org">sfowler@vipnet.org</a>	804.786.6220
Arnold, Billy	VIPNet	<a href="mailto:warnold@vipnet.org">warnold@vipnet.org</a>	804.786.6204
Boehm, Deanna	VIPNet	<a href="mailto:dboehm@vipnet.org">dboehm@vipnet.org</a>	804.786.4230

Name	Organization	E-mail	Phone
Willet , Rodney	VIPNet	<a href="mailto:rod@vipnet.org">rod@vipnet.org</a>	804.786.6202
Neudeck, Dave	VIPNet	<a href="mailto:dave@vipnet.org">dave@vipnet.org</a>	804.692.0574

### Web Services Workgroup Support

Name	Organization	E-mail	Phone
Carter, Leslie	Department of Information Technology	<a href="mailto:lcarter@dit.state.va.us">lcarter@dit.state.va.us</a>	804.371.5577
Hughey, Hal	Department of Information Technology	<a href="mailto:hhughey@dit.state.va.us">hhughey@dit.state.va.us</a>	804.371.5785
Williams, Chris	Department of Information Technology	<a href="mailto:cwilliams@dit.state.va.us">cwilliams@dit.state.va.us</a>	804.371.5674
Akers, Janice	Department of Information Technology	<a href="mailto:jakers@dit.state.va.us">jakers@dit.state.va.us</a>	804.786.1153
Maxwell, Ted	Department of Labor & Industry	<a href="mailto:tedmaxwell@doli.state.va.us">tedmaxwell@doli.state.va.us</a>	804.786.1397
Lubic, Paul	Department of Technology Planning	<a href="mailto:plubic@ntp.state.va.us">plubic@ntp.state.va.us</a>	804.371.0004
Perkins, Eric	Department of Technology Planning	<a href="mailto:eperkins@ntp.state.va.us">eperkins@ntp.state.va.us</a>	804.786.0516
Zenreich, Alan	Ness Technologies	<a href="mailto:alan.zenreich@ness-usa.com">alan.zenreich@ness-usa.com</a>	201.488.7222 X 160
Stock, Doug	Ness Technology	<a href="mailto:doug.stock@ness-usa.com">doug.stock@ness-usa.com</a>	703.464.0133
Carver, Elaine	Roanoke County	<a href="mailto:ecarver@co.roanoke.va.us">ecarver@co.roanoke.va.us</a>	540.561.8003
Graham, Ellen	Software AG, Inc.	<a href="mailto:ellen.graham@softwareagusa.com">ellen.graham@softwareagusa.com</a>	603.487.2039
Reagan, Tana	Software AG, Inc.	<a href="mailto:tana.reagan@softwareagusa.com">tana.reagan@softwareagusa.com</a>	703.391.6929
Hunter, Jenny	SoTech/COTS	<a href="mailto:jhunter@gov.state.va.us">jhunter@gov.state.va.us</a>	804.786.9579
Smith, Harry	Department of Education	<a href="mailto:hsmith@mail.vak12ed.edu">hsmith@mail.vak12ed.edu</a>	804.225.2951

### Guests in Attendance

Name	Organization	E-mail	Phone
Ambs, Chris	Software AG, Inc.	<a href="mailto:chris.ambs@softwareagusa.com">chris.ambs@softwareagusa.com</a>	201.461.1730
Mason, Brian	Department of Technology Planning	<a href="mailto:bmason@ntp.state.va.us">bmason@ntp.state.va.us</a>	804.371.0007
Norman, Fred	CVC	<a href="mailto:fred.norman@cvconlin.net">fred.norman@cvconlin.net</a>	804.690.1497

## Table of Contents

<b>Executive Summary.....</b>	<b>9</b>
Vision and Mission.....	9
An Introduction to Web Services .....	9
Web Services Workgroup Goals and Objectives .....	10
Findings and Conclusions .....	11
Recommendations .....	13
Team Members and Resources.....	14
 <b>Introduction.....</b>	 <b>15</b>
Web Services Defined .....	15
Objectives of the Workgroup .....	18
General Assumptions for the Proof-of-Concept.....	18
 <b>Overview of the Web Services Workgroup Process.....</b>	 <b>20</b>
Goal for Creating a Successful Web Service Proof-of-Concept.....	20
Web Services Diagram: .....	21
Project Teams .....	22
Project Management Approach.....	25
Function Selected for the Web Services Proof-of-Concept.....	27
Proof-of-Concept Schedule .....	29
Web Service Components for Change of Address.....	29
UDDI Design and Requirements .....	30
Design Issues Encountered and Resolved .....	34
Test Data.....	39
Security for First-Generation Web Services .....	39
Training.....	42
 <b>Best Practices .....</b>	 <b>45</b>
Lessons Learned from Industry .....	45
Best Practices Derived from the Project.....	45
 <b>Tools and Resources .....</b>	 <b>48</b>
Tools.....	48
Resources .....	53
Level of Effort.....	53
 <b>Appendices .....</b>	 <b>55</b>
 <b>End Notes.....</b>	 <b>183</b>



## EXECUTIVE SUMMARY

### VISION AND MISSION

The Charter for the COTS Web Services Workgroup clearly defined the following vision and mission for the Proof-of-Concept project.

#### **Vision**

- Within the established schedule, the objective of this workgroup was to have evaluated Web Services and supporting technologies that promote the development of sharable applications accessible via the Internet and/or organizational Intranets.

#### **Mission**

- Determine the viability of Web Services as an application development and integration approach that will create a secure, standard-based platform independent framework for interoperability that supports activities of the commonwealth.

### AN INTRODUCTION TO WEB SERVICES

Web Services are self-contained business functions that operate over the Internet. They are written to strict specifications to work together and with other similarly constructed components. Some of the more established functions at this stage are messaging, directories of business capabilities, and descriptions of technical services. But other functions are progressing as well.

Web Services are important to business because they enable systems in different organizations to interact more easily with each other. With businesses needing closer cooperation between suppliers and customers, engaging in more joint ventures and short-term marketing alliances, pursuing opportunities in new lines of business, and facing the prospect of more mergers and acquisitions organizations need the capability to link-up their systems quickly with other companies. Thus Web Services gives commonwealth agencies, institutions of higher education, and localities the capability to do more business electronically, with more potential business partners.

Due to Web Services being written according to standards sanctioned by the W3C<sup>1</sup> and the OASIS<sup>2</sup>, all

---

<sup>1</sup> "The World Wide Web Consortium exists to realize the full potential of the Web.

The W3C is an industry consortium that seeks to promote standards for the evolution of the Web and interoperability between WWW products by producing specifications and reference software. Although industrial members fund W3C, it is business partner-neutral, and its products are freely available to all. The Consortium is international; jointly hosted by the MIT Laboratory for Computer Science in the United States and in Europe by INRIA who provide both local support and performing core development. The W3C was initially established in collaboration with CERN, where the Web originated, and with support from DARPA and the European Commission."

<sup>2</sup> OASIS (Organization for Structured Information Standards) is a nonprofit, international consortium whose goal is to promote the adoption of product-independent standards for information formats such as Standard Generalized Markup Language (SGML), Extensible Markup Language (XML), and Hypertext Markup Language (HTML). Currently, OASIS (formerly known as SGML Open) is working to bring together competitors and industry standards groups with conflicting perspectives to discuss using XML as a common Web language that can be shared across applications and platforms.

parties work from the same basic design. Organizations then add value and business advantage to the basic design to meet the needs of their customers. For example, an organization can offer its suppliers the capability to view inventory levels of products the suppliers provide so they can replenish the stocks without the customer cutting separate purchase orders. Web Services provide the basic messaging and service-description functions for this kind of electronic relationship. However, suppliers could build on these basic features to provide better services to the customer and allow organizations to extend their capabilities to other trading partners.

Since Web Services are built on standards, they make it possible for many systems developers to enter the market, which increases competition and (theoretically) reduces costs. The competition among business partners also encourages more innovation in the products and services offered to business customers. Basing systems on standards helps prevent being locked-in to a specific business partner or type of computer or software.

## **WEB SERVICES WORKGROUP GOALS AND OBJECTIVES**

To understand the promises as well as the current level of Web Services development, the workgroup, undertook six separate Proof-of-Concept initiatives based on “change of address.” This project involved research and development activities related to Web Services and supporting technologies of agencies and business partners.

The Charter for the COTS Web Services Workgroup clearly defined the following goals and objectives for the Proof-of-Concept project:

### **COTS Strategic Goals**

- To assist the Secretary of Technology to monitor trends and advances in fundamental technologies of interest and importance to the economy of the Commonwealth of Virginia.

### **Workgroup Objectives**

- To determine the probable impact of Web Services technology on the strategic direction of application development and integration in the private and public sectors.
- Through a technical R&D process, determine the feasibility, advantages, disadvantages, and best practices when implementing Web Services in the Commonwealth of Virginia.
- To determine requirements for implementation that will include (but will not be limited to) hardware, software, training, and security and migration issues. Also, to determine potential cost scenarios.
- To make a recommendation regarding adoption of Web Services technology for the Commonwealth of Virginia.

The Workgroup approached the project with the following perspectives in mind. Technical capabilities should not be a solution looking for a problem. Business problems must look for technical solutions. Technology can be implemented incrementally. Technology implemented consistently and commonly

among collaboration participants brings great value, even for less glamorous technology implementations. Technology should open more doors than it closes. The use of open standards should not lock participants into a business partner, a product, or another participant's capabilities. Technology support should not exceed the skills of those responsible for support. On the other hand, this reality should not prevent the selection of advanced solutions, but it should be considered in developing the implementation strategy. Efforts should not be duplicated if at all possible. Instances of competing standards within the framework should be minimized. In creating standards, it is better to borrow than build. Standards, needed among participants to make systems interoperable, should be selected by consensus, and be implemented incrementally to ensure they meet the requirements of the commonwealth.

## FINDINGS AND CONCLUSIONS

1. As currently approved by the W3C and OASIS groups, and as implemented in the market, Web Service specifications for XML, SOAP, WSDL and UDDI appear to fundamentally work "as designed." The workgroup was able to create a network of loosely coupled, reusable services that communicated and cooperated with each other. The Proof-of-Concept used a sufficiently complex business problem that forced the need for multiple Web Services to communicate with each other, as well as interact with middleware and legacy systems, thus going far beyond the creation of a simple Web Service where (typically) a single person interacts with a single service in a web-based setting. While problems were encountered, most were overcome through improved team communication (exchange of technical information) and minor to moderate application changes. Only one product "patch" was required throughout the project.
2. The specifications for Web Services are not mature. In fact, they are a constantly moving target and this makes Web Services difficult for people to understand technically and feel comfortable with in terms of product investment. It should be noted that much of this change is focused on UDDI and WSDL. While XML and SOAP are also undergoing change, these specifications are not seeing as much flux. By far, though, the most change being introduced involves new/additional specifications such as the Security Assertion Markup Language (SAML) and the Business Process Execution Language for Web Services (BPEL4WS). Many organizations (both public and private) have opted to wait on the sideline for this period of change to slow down. Others have opted in, and have begun making investments, primarily in the low to moderate risk development projects. These individuals seem to have acknowledged the foothold established by XML, SOAP, WSDL and UDDI and view all this "change" energy as obvious market acceptance. They have also accepted the fact that until more enhancements are made to the specifications they will live with the deficiencies until either (1) the standards they need emerge and are converted into usable products or (2) internally crafted or business-partner supplied solutions emerge to temporarily fill in the gaps.
3. Web Services technology is not inherently "fast" to implement. While it is true that reports have been made claiming a 10-15 percent reduction in development time using Web Services to integrate applications (versus a full or partial re-write) it is also true that the speed of development is more a factor of how much human collaboration occurs and the existence of well-constructed and easy-to-use development toolkits that insulate the developers from the minutia of the architecture itself. So, from that angle, Web Services is currently "business as

usual." However, Web Services will appreciably accelerate the development process in a sustained manner and reduce time-to-deployment once an organization has begun to establish a cache of reusable "services." In general, this cache will evolve across most organizations. First, the services will support fairly simple business processes and, over time, they will become significantly more complex as more coupling occurs and, as a result, reuse will show significant reductions in the development cycle.

4. Some degree of centralized management for UDDI directories and the WSDL definitions they contain seems appropriate (this is above the notion of a centrally managed project). "Consuming" a service begins with first "discovering" it. This is the primary function of the UDDI specifications. If there are errors in the directory, confusion regarding how it is constructed, or difficulties accessing it, then the services it lists are not going to be available. UDDI implementation and ongoing maintenance requires a high level of control.
5. Interoperability and integration are not completely facilitated by Web Services technology, but they help significantly. Legacy systems of all kinds (by definition) exist behind the Web Services "layer". This means that there must be something that unites these two environments to actually realize interoperability and integration. That solution today is comprised of various forms of middleware, a software domain in which the commonwealth already has a significant investment. Whether it is internally developed code running on a Windows IIS server that exchanges information with an Oracle based system, or a purchased package that supplies various communication adapters that can exchange data with Unisys and CICS based systems -- something must exist that will allow a web service to drive the code that actually makes business occur. A value gleaned from Web Services is that these environments (with standards-based development) can now find each other and communicate using the existing infrastructure of the Internet.
6. It is unclear whether the current Web Services specifications will be enhanced to include specialized functions that will allow organizations to efficiently "manage" complex services. One proposed standard, Open Management Interface (OMI) does exist, but it has not been approved by the W3C or OASIS and moved into product development by the industry. Currently, it seems that not much attention is going into that area which basically says that development teams (and technical support teams they rely upon) are going to have to know completely the pieces and parts of each service they deploy. Otherwise, inevitable failures and disruptions to service may be difficult to correct. The bottom line is that they'll not get much help from a rich set of diagnostic tools in the near future. That same condition did not appear to slow the adoption of the Internet in the past decade, but it does seem to imply a requirement for significant commitment to technical training.
7. Web Services, as high-tech and bleeding-edge as it may be, continues to require a significant investment in people. More specifically, coordination and communication between people. To successfully automate a business process by "consuming" reusable services owned by other entities (inside or outside an organization) requires close collaboration. Without it, expectations will not be met and change management will be extraordinarily difficult if not impossible. So, within an organization or among a group of collaborating organizations, strong project management skills will reap big rewards.

8. No firm direction has been set in the private or public sector in terms of a cost model (or how to charge) for "consuming" Web Services. At this point, there is no emerging W3C or OASIS standard that the workgroup knows about nor any de facto approach sanctioned by the market.
9. Web Services technology, at present, can make use only of security specifications and products on the market today. While a tremendous amount of energy is being put into new security specifications tailored for Web Services, it is accurate to state that conventional approaches involving such options as Secure Sockets Layer (SSL), IP address checking, digital signatures/server certificates, ID/password and traditional PIN checks are really what are available. Is this enough for low-to-moderate risk, less complex Web Services implementations? Yes. It is certainly no less secure than what is broadly in place today and used for many value-based transactions. Is it enough for situations where both ends of the service are from known, pre-established IP addresses (locations)? Yes. However, to accommodate complex business processes using multiple Web Services, more robust security that provides sophisticated policy-setting, authentication, and trust algorithms in a highly distributed, transaction-based environment must be in place. It is also useful to mention the need for sophisticated standards that address transactional integrity. One year from now, the security options for Web Services will be significantly different.

## RECOMMENDATIONS

Based upon the research and development activities of the workgroup, the following assertions are made:

- . . .Web Services technology, while not mature, is indeed viable and a strategic next-step for developing platform-neutral web-based applications (services) that can be shared and/or reused by entities capable of communicating using the proper specifications and protocols.
- . . .Web Services technology does provide a critical "link" in terms of facilitating application interoperability and integration in a world where business processes have come to depend upon both web and legacy environments.
- . . .The commonwealth's significant commitment and impressive progress towards extending services to the citizens and businesses of the state via the web position it well to make use of this new technology. If implemented, Web Services will significantly improve business process efficiency and performance through improved application interoperability and integration.

Based upon the above findings, conclusions, and assertions, the workgroup makes the following recommendations:

1. COTS should establish a workgroup that will define procurement criteria for products that create, implement, and maintain Web Services technology. Upon completion, and given a compatible financial climate, the proper entities should then initiate formal procurement activities. It is further recommended that initial Web Services deployments focus on low-to-moderate risk business processes and application (as determined by appropriate management

and relevant policies) and provide for thorough staff training to ensure adequate support.

2. The Council on Technology Services (COTS) should consider the merits of becoming actively involved with relevant technical committees of the W3C and OASIS organizations. This would improve the commonwealth's ability to influence future proposed standards (or changes) and evaluate technological developments very early in their life cycle.
3. COTS should request the Department of Technology Planning (DTP) to begin reviewing all relevant policies, standards and guidelines for possible changes and enhancements that will accommodate and properly channel the use of Web Services technology. COTS members (non-state) with similar internal organizations should be encouraged to initiate the same process.

## TEAM MEMBERS & RESOURCES

The workgroup comprised six change-of-address Proof-of-Concept teams in addition to a UDDI team. Each team consisted of public and private sector personnel. Seven state agencies, one institution of higher education, one county and city, and six private-sector software providers participated in the project (several other private-sector software integrators were added as the project progressed). Teams ranged from four to 13 individuals. Many team members were geographically spread out across the commonwealth. In most cases the leader was a senior public sector IT manager supported by senior private sector personnel as well as other public sector associates. The Workgroup spent seven months and a total of 2,480 hours on the project, which averaged out to 354 hours per team.

## INTRODUCTION

### WEB SERVICES DEFINED

#### What Web Services Really Are<sup>i</sup>

Once the hype is eliminated, Web Services are quite basic. They are modular software components wrapped inside a specific set of Internet communications protocols. Here's what they can do:

- Connect applications that are addressable via the Internet,
- Enable applications to communicate automatically without human intervention,
- Help facilitate the automation of business processes that make use of web-addressable application and involve multiple non-web (legacy) platforms,
- Be deployed for use over the Internet, on an intranet inside a corporate firewall, on an extra-net, or a combination of these three environments, and;
  - Be written using a wide variety of development tools.

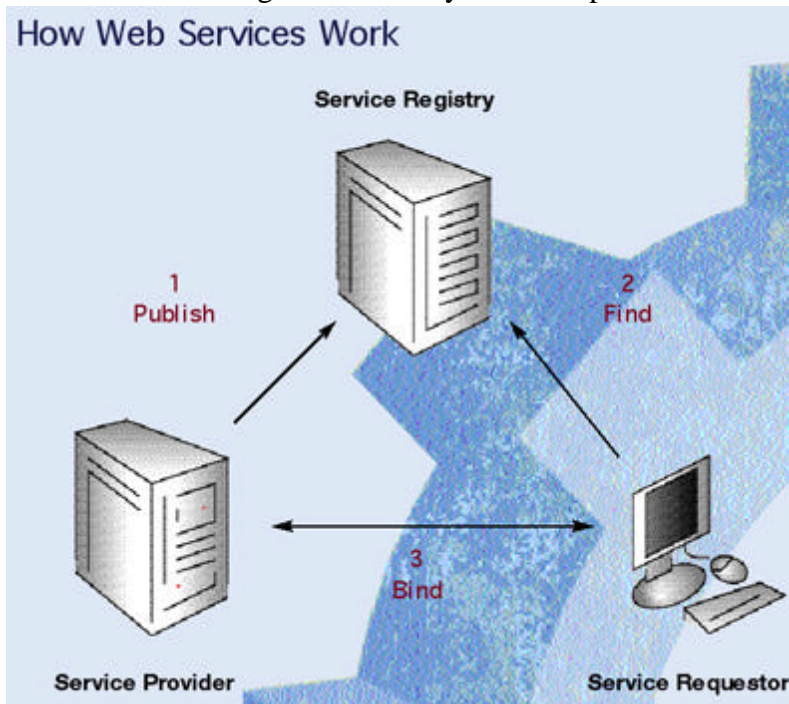


Figure 1, How Web Services Work

---

*"To paraphrase Mark Twain: 'Everyone talks about Web Services, but no one really knows what they are,'"*

Larry Marion, editor-in-chief of  
EnterpriseSoftwareHQ.com

---

#### Web Services:

The Web Services Architectural Requirements Working Group of the World Wide Web Consortium (W3C), which develops standards for interoperable technologies, defines Web Services to be a software application identified by a URI, whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via internet-based protocols.

## Understanding the Key Protocols

There are four key protocols that make Web Services possible:

1. Extensible Markup Language<sup>ii</sup> (XML)-structures information so that data can be easily extracted and used by other applications. An XML document describes a web service and includes information detailing exactly how that web service can be run. When someone runs a web service, the XML document is found first. Then, the document details how to run the service.
2. Web Services Description Language (WSDL)-used to create the XML documents that describe a web service.
3. Universal Description, Discovery and Integration (UDDI)- allows for the creation of public, searchable directories of Web Services. Many Web Services will ultimately be located in public directories so that anyone can search for a particular service and then run it, based on the XML document describing the service.  
([www.uddi.org/find.html](http://www.uddi.org/find.html) provides a look at an early UDDI implementation of these directories)
4. Simple Object Access Protocol (SOAP)-underlies everything else. It's a communications protocol that allows structured data such as XML to be exchanged between network applications.

## Where Web Services Are Today

Web Service protocols and specifications have gained widespread support among technology providers. Industry leaders such as BEA, Microsoft, SilverStream, Software AG, Sun, webMethods, HP, and IBM all support the standards and have released toolkits, hardware and software for developing and deploying Web Services. Integration firms like webMethods of Fairfax, Va., can turn existing applications and business processes into Web Services. Security firms like RSA Security of Bedford, MA, offer tools providing security for Web Services. Companies such as Bowstreet of Portsmouth, N.H., have created platforms to help build and manage web services-based applications. The bottom-line is that many companies have incorporated the Web Services specifications into their product-line.

Growing interest and rapid adoption have also generated some challenges. One problem is that the underlying standards are still evolving; so different business partners are forced to provide proprietary solutions for areas of the specifications that are simply not mature (for example, workflow). Also, development toolkits and many of the applications they produce continue to have OS and hardware platform dependencies. While the Web Services protocols do enable platform-neutral communication across the web, the applications themselves still may need certain supporting hardware and software to execute properly-not exactly a new problem in the world of application development, but it is worth noting that Web Services does not

---

### Web Services' Underlying Protocols: A Glossary of Terms

- **XML (Extensible Markup Language):** A markup language that structures information so that the information can be easily extracted and used by other applications. It uses tags, as does HTML, but those tags are used to structure and define information rather than display it. Service descriptors that detail how Web Services can be located and run are written in XML.
- **SOAP (Simple Object Access Protocol):** The protocol through SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses. SOAP can potentially be used in combination with a variety of other protocols; however, the only bindings defined in this document describe how to use SOAP in combination with HTTP and HTTP Extension Framework.
- **WSDL (Web Services Description Language):** The language used to create service descriptions of Web Services. It can be used to describe the location of the service and how to run it, as well as what business is hosting the service, the kind of service it is, keywords associated with the service and similar information.
- **UDDI (Universal Description, Discovery and Integration):** The directory technology used by service



fix or address that issue. There is also a cultural issue with Web Services improving the ability to automate processes within and across organizations, how far will the people that are accountable for those processes be willing to take this new technology? Will there be a point where they feel they have lost control?

Pete Lindstrom, director of security strategies at the Framingham, MA-based Hurwitz Group, says these kinds of short-term problems are inevitable, given the nature of what Web Services is trying to accomplish. "In a lot of ways, Web Services is all about the interoperability problems we have today between systems," he points out. Thus, it is only natural that at first there would be these kinds of issues.

Most observers believe these issues are nothing more than the normal growing pains all new technologies face. Given how new and complex the standards are, they say we have come further along in a shorter amount of time than one would expect.

"Those issues will end up being resolved," predicts EnterpriseSoftwareHQ.com's Marion, "There's too much at stake for everyone involved. Once everyone realizes the benefits offered by Web Services, even cultures inside organizations will change to accommodate building them."

John Worrall, vice president of product marketing for RSA Security, sums up many people's thoughts about the future of Web Services: "I believe we have not yet tapped into the full power of what the Internet can do for us. Web Services will show us the way."

### **Web Services Are Positioned for Rapid Growth**

Although still in their infancy, Web Services is rapidly attracting the attention of the business world with their promise of easy information exchange, reduced programming costs, improved connectivity and collaboration with partners and customers, and more. In a survey of businesses by the research firm TechMetrix<sup>iii</sup>, 48 percent of managers said they were either in the testing and prototype stage or had already started Web Services projects while 32 percent expressed interest in Web Services (See chart below). A survey of almost 500 IT managers polled in 2001 by The Business Integrator Journal found 65 percent of respondents reporting that Web Services are strategically important to IT.

Gartner Group<sup>iv</sup> estimates a \$21 billion market worldwide by 2005 for software that uses Web Services standards. ZapThink, a research firm focusing on XML, projects the Web Services market to grow from \$380 million in 2001 to \$15.5 billion in 2005.

registries that allows the directory to be searched for a particular Web service. In essence, UDDI is a White Pages or Yellow Pages that can be used to locate Web Services. There can be both private and public UDDI directories.

---

### **How Web Services Work**

Web Services are software modules wrapped inside a specific set of Internet communications protocols and can be run over the Internet. In order for the Web service to be run, it needs to be described in detail so that other programs can understand what it is and know how to connect to it. It is described using the Extensible Markup Language (XML). This XML depiction is called a service description or descriptor and includes all the details necessary for the Web service to be accessed.

For a computer or program to use a Web service, it needs to be able to find this service description and then bind to it. To accomplish this, there are three key roles in the Web Services architecture: Service provider: The server that hosts the Web service; Service registry: A searchable database that hosts service descriptions; and Service requestor: The person, computer or service looking to run a Web service.

Together, they perform three operations on a Web service:

1. **Publish**-The service provider hosting the Web service module creates an XML-based service description for the Web service. It uses the publish operation to make information about the service available so that it can be found and used.
2. **Find**-The service registry, using the UDDI protocol, makes service descriptions available so that Web Services can be found and run. A computer or program can search for, and understand, what the Web service is, where it's located

## OBJECTIVES OF THE WORKGROUP

As stated in its charter, the Web Services Pilot Project workgroup supports the Council on Technology and Science (COTS) strategic goals. The alignment of COTS strategic goals and workgroup objectives is shown below. The workgroup's work plan developed as a result of this charter will, at a minimum, establish measures of success for each objective and provide for regular workgroup performance reporting to COTS.

<b>COTS Strategic Goals</b>	<b>Workgroup Objectives</b>
To assist the Secretary of Technology to monitor trends and advances in fundamental technologies of interest and importance to the economy of the commonwealth.	<p>To determine the probable impact of Web Services technology on the strategic direction of application development in the private and public sectors.</p> <p>To determine the technical feasibility, advantages, disadvantages and best practices when implementing Web Services in the commonwealth.</p> <p>To determine a cost model.</p> <p>To make a recommendation regarding adoption of Web Services technology for the commonwealth.</p>

Figure 2, Workgroup Goals & Objectives

- and how to link to it.
3. **Bind and run the service-**  
After the service requestor finds the Web service's description, it has the information it needs to run the Web service.

## GENERAL ASSUMPTIONS FOR THE PROOF-OF-CONCEPT

- Teams agreed to change address scenarios where each team provided functionally to change addresses in a system that was representative of, or integrated with, the systems of the agency/localities. As the change was made it was then pushed out to the other teams utilizing Web Services technologies.
- The group agreed to only perform address changes. They would not add or delete addresses or use phone numbers.
- Teams agreed to a common XML address schema.
- Teams agreed to use a common originating Web Services description (WSDL) file and to implement three services. 1) Get address, 2) Change address, and 3) Query address changes.
- Teams agreed to use data and systems that adequately represented the agency/locality each business partner was paired with.

- Teams agreed to use back-end systems where possible. If they could not or it was not feasible, the teams simulated the needed platforms and software.
- Teams agreed to have all services available on the public Internet and have them registered in a public UDDI repository to be housed by VIPNet.
- Teams agreed to keep security to a minimum since the Web Service specifications in this area are not well developed and formally approved in many cases. The reality is that current Web Service technology depends upon existing web security standards such as SSL and proprietary authentication and authorization methodologies.

## OVERVIEW OF THE WEB SERVICES WORKGROUP PROCESS

### GOAL FOR CREATING A SUCCESSFUL WEB SERVICE PROOF-OF- CONCEPT

1. Technology is not the Primary Driver for Solutions – The availability of a technology should not drive collaboration initiatives. Technical capabilities should not be a solution looking for a problem. The converse is true – business problems must look for technical solutions, and must be willing to invest in technology when necessary.
2. Technology can be Implemented Incrementally – Participants must recognize the value of incrementally delivering capabilities. XML standards can be valuable long before data warehousing capabilities become beneficial. Firewalls can provide security before role-based authentication is available. Technology implemented consistently and commonly among collaboration participants brings great value, even for less glamorous technology implementations.
3. Technology Should Open more Doors than it Closes – Technology solutions should interact and interface using open standardized solutions. These will not lock participants into a business partner, a product, or another participant's capabilities. Collaboration should create choices, not dependency. Exceptions should be rare.
4. Technology Support Should not Exceed Available Skills – Technology choices should not exceed the skills sets of those responsible for support. A clever solution in one jurisdiction may work well for them, but may exceed the capabilities of participants in other jurisdictions. This reality does not prevent the selection of advanced solutions, but it should be considered in developing the implementation strategy.
5. Efforts Should not be Duplicated if at all Possible – In creating standards, it's better to borrow than build. Instances of competing standards within the framework are to be minimized. If an accepted XML-based standard can be used in a government service, then it should be used rather than creating an additional stove-piped language for data exchange.

---

*“Web Services do for applications what hypertext mark-up language, HTML, does for content,”*

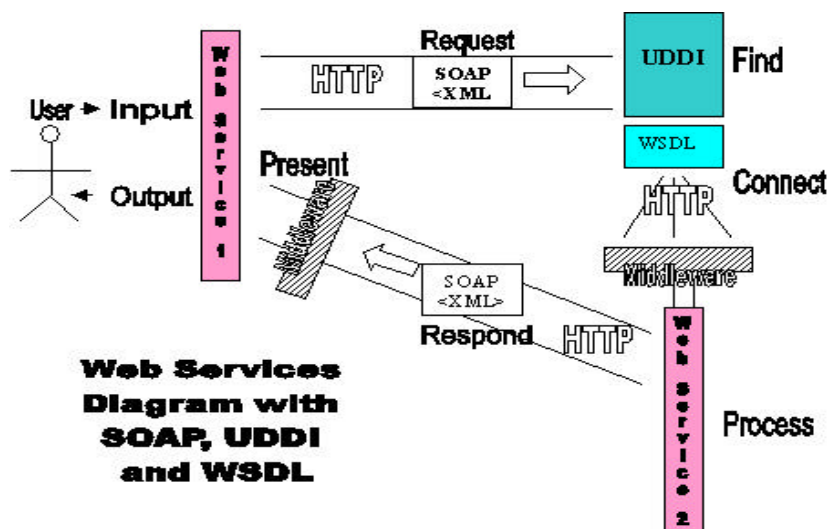
Ted Schadler, Group Director,  
Forrester Research in  
Cambridge, Mass.

6. Strive toward Voluntary Standards – Standards are needed among participants to make systems interoperable. However, in the inter-governmental environment these standards cannot be unilaterally imposed. Standards should be selected by consensus, and be implemented incrementally to ensure they meet the requirements of the commonwealth.

#### WEB SERVICES DIAGRAM:

The Web Services diagram below illustrates the dynamic approach to Web Services. Following the UDDI model, WSDL (Web Services Definition Language) could be used to extract and execute all defined Web Services. This would save developers from having to know and hard code the individual web service URL's for each of the participating groups. This is especially important since those URL's could change as often as servers change. It's also important because as the Web Services registry grows across services and jurisdictions, metadata will be needed to search for and identify through an exponentially increasing number of Web Services.

In the future, as XML becomes more widely used in government, participants would want to develop a web service request/response mechanism that can be securely transported via Simple Object Access Protocol (SOAP) for purely dynamic data exchange rather than doing it manually. This prospect could closely resemble the UDDI standards under development.



#### URL (Universal Resource Locator)

A standard way of specifying the location of an object, typically a web page, on the Internet. Other types of object are described below. URLs are the form of address used on the World-Wide Web. They are used in HTML documents to specify the target of a hyperlink, which is often another HTML document (possibly stored on another computer).

## PROJECT TEAMS

### Team Makeup:

**Team 1** – Department of Motor Vehicles (DMV), Microsoft Corporation, Susquehanna Technologies

**Team 2** – Department of Alcohol and Beverage Control (ABC), BEA Systems, Inc.

**Team 3** – Department of Human Resources Management (DHRM), Virginia Retirement System (VRS), SilverStream, Inc.

**Team 4** – County of Roanoke, City of Virginia Beach, Sun Microsystems, Inc.

**Team 5** – Department of Medical Assistance Services (DMAS), Department of Education (DOE), Software AG, Inc.

**Team 6** – George Mason University (GMU), WebMethods, Inc.

**Team UDDI** – Virginia Information Providers Network (VIPNet) and BEA Systems, Inc.

*Please refer to page 30 for Team UDDI's roles, responsibilities and resources.*

### Team Roles and Responsibilities:

**Team 1** – Department of Motor Vehicles (DMV), Microsoft Corporation, Susquehanna Technologies

Business/Functional Requirements:

- Provide an address change function to the citizen, which will use Web Services in the background to update subscribers of the service of the address change information.
- Address change information will be provided (pushed) to all subscribers in a change address function, if the change originated at the DMV.
- A get address function will be provided, if a subscriber chooses to retrieve (pull) the address information.
- A query address changes function will also be provided. A Web Service subscriber may choose to retrieve (pull) all address changes performed between specified date/time ranges for a specified agency/locality (originator).

---

*"We were pleased with the maturity of the development toolsets available, given the relative infancy of Web Services related technologies. We were able to quickly build the components' foundation through the tools provided, allowing us to concentrate on the functionality and behavior of the application, rather than the low-level architecture."*  
- Bill Vencil, Susquehanna Technologies (Team 1)

- Only address changes that originated at the DMV will be propagated to all subscribers. This will prevent recursive looping of address change processing with other agencies/localities.

**Team 2** – Department of Alcohol and Beverage Control (ABC), BEA Systems, Inc.

Business/Functional Requirements:

Service Requirements:

- The system will provide a subscription service that will take information pertaining to a type of address, a form of update mechanism (i.e. push or notify), and a callback URL (WSDL).
- The system will get subscribers, input address type, list of update type (push or notify) and a callback URL (WSDL).

General Requirements:

- The system must invoke to subscribe noting the address type, update type, and callback URL.
- The system must get subscribers, their address type, a return list with the update mechanism (push or notify), and a call back URL.
- The system must invoke the callback URL (WSDL) for a notify.
- The system must invoke callback ULR (WSDL) for a push.
- The system must contain a method to get address type, which will indicate if the type is an individual or business and a unique identifier.
- On notify, the system must indicate the type, individual business, and a unique identifier.
- On push, the system must indicate the type, individual or business, and the data.

Functional/Technical Assumptions:

- The system will assume all business partners/contacts have Virginia addresses for the purpose of this prototype.
- The system will assume that a company address change will be applied to all contacts for the given company.
- Design will begin under the assumption that the group will determine the address schemas that will be published.

**Team 3** – Department of Human Resources Management (DHRM), Virginia Retirement System (VRS), SilverStream, Inc.

Business/Functional Requirements:

- Implement Proof-of-Concept of Web Services on Change of Address representing the following agencies: Virginia Retirement System –

---

*"While the workgroup's findings were very informative regarding Web Services interoperability, the COTS workgroup "forum" itself proved also to be extremely valuable. Multiple competing business partners working together cooperatively and successfully with the sole purpose of evaluating technology for the Customer, a scenario unlikely seen in the commercial sector".*

- Dan Lender, BEA Systems, Inc. (Team 2)

---

*"Software design is always a very important part of software development. But it becomes more evident in implementing the technology of Web Services. Without an extremely well thought out design and design process, a web services application can get exponentially more complicated than is required."*

- Tanzeena O'Brien, SilverStream Software, Inc. (Team 3)

change home and mailing address and Department of Human Resources Management – change home address

- Implement Proof-of-Concept of Web Services on Get Address representing the following agencies: Virginia Retirement System – get home and mailing address and Department of Human Resources Management – get home address
- Change and get address enablement of the VRS system through a 3270 interface.
- Change and get address enablement of the DHRM system through the Mitem interface.
- Distribute a Change of Address request to the other agencies participating this Proof-of-Concept.

---

*"This was a great project to work on and everyone that touched it has a better understanding of the value of Web Services."*

- Wayne Cox, Sun Microsystems, Inc. (Team 4)

**Team 4** – County of Roanoke, City of Virginia Beach, Sun Microsystems, Inc.

Business/Functional Requirements:

- Implement the Web Services Change of Address Proof-of-Concept representing Roanoke County and Virginia Beach. Distribute Change of Address requests to the other teams participating in the Web Services Proof-of-Concept

---

*"We look forward to the maturation of Web Services technology. Web Services has tremendous potential to influence the way we do business in the commonwealth."*

- Bethann Canada, Virginia Department of Education (Team 5)

**Team 5** – Department of Medical Assistance Services (DMAS), Department of Education (DOE), Software AG, Inc.

Business/Functional Requirements:

- Implement the Web Services Change of Address Proof-Of-Concept representing the Department of Education and the Department of Medical Assistance Services for two types of Addresses: Home and Business
- Represent the Web Service “enablement” of DOE and DMAS systems currently residing in Oracle DBs
- Implement the distribution of Change of Address requests to the other teams participating in the Web Services Proof-Of-Concept

---

*".... the resulting Proof-of-Concept application demonstrated the feasibility and potential value of Web Services to the higher education community".*

- John Creuziger, George Mason University (Team 6)

**Team 6** – George Mason University (GMU), WebMethods, Inc.

Business/Functional Requirements:

Business Requirements:

- Complete Proof-of-Concept address change application
- Demonstrate the feasibility of integrating applications with SCT Banner
- Realistically determine staff and training investments
- Work within the computing architecture of GMU



#### Technical Requirements:

- Use webMethods products to develop/host application
- Use XML, SOAP, WSDL, and UDDI
- Use Java2 technology suite
- Use SSL
- Use Oracle database

#### Functional Requirements:

- Authenticate requestor via ID/PIN combo
- Prompt for pertinent address type(s)
- Display old address info/prompt for new
- Update appropriate local addresses
- Route request to each development team
- Process Web Services updates from other teams
- Log all address change activity (or lack)
- Generate confirmation to requestor

#### Assigned Components by Teams

The workgroup members identified the following core components of the Proof-of-Concept address change application:

Team #	Gov't Entity	Business Partner	Home	Garage	Mailing	Business	Billing	Security
1	DMV	Microsoft	X	X	X			X
2	ABC	BEA			X	X		X
3	DHRM & VRS	SilverStream	X		X			X
4	Va. Beach & Roanoke	Sun Microsystems	X			X	X	X
5	DOE & DMAS	Software A G	X			X		X
6	GMU	webMethods	X				X	X

**NOTE:** VIPNet & BEA Collaborated to develop the UDDI component

Figure 3, Assigned Components by Team

## PROJECT MANAGEMENT APPROACH

### Reporting Requirements

To keep reporting by the six teams consistent and to simplify the process a template was developed and utilized by each team. Each team made two status reports which were rolled into their final report. A copy of each teams

final report is contained in the appendix to this document. All team report findings are published in the Findings and Conclusion section (VII), and summarized and included in the Executive Summary section (I) to the Web Service Proof-of-Concept report.

### **Template**

Each team in preparing their reports utilized the following template and instructions.

<b>1. PROJECT PLAN (SCHEDULE)</b>	Attach high-level project plan with list of meetings, time and location
<b>2. LIST OF TEAM NAME, MEMBERS, ROLES</b>	Statement of team member roles and responsibilities – who is doing
<b>3. PROOF-OF-CONCEPT DESIGN</b>	Attach general design of your application narrative, flow diagrams,
<b>4. BUSINESS/FUNCTIONAL REQUIREMENTS</b>	Statement of the specific business/functional requirements
<b>5. PROJECT EQUIPMENT DESCRIPTION</b>	Full description of the hardware and software to be used, who is
<b>6. ACQUISITION AND INSTALLATION</b>	Indicate whether additional hardware and software were purchased or
<b>7. SERVER REQUIREMENTS</b>	Describe changes to existing server or acquisition that will be needed
<b>8. NETWORKING REQUIREMENTS</b>	Describe changes to existing network or procurement that will be needed
<b>9. TRAINING</b>	Identify the technical and business training that has been or will be

<b>REQUIREMENTS</b>	needed to prepare staff for proof-of-concept project.
<b>10. TESTING PLAN</b>	Attach a list showing all elements to be tested. This should include testing and validation of all elements of the proof-of-concept.
<b>11. DEVELOPMENT/TECHNICAL</b>	High-level description of what it was like to use the tools in the development of the proof-of-concept project.
<b>12. PROTOCOL/SPECIFICATION ISSUES</b>	XML, SOAP, WSDL, and/or with UDDI
<b>13. INTEROPERABILITY ISSUES</b>	Describe in detail.
<b>14. OTHER CONCERNS/ISSUES</b>	Team concerns/issues (technical, organizational, logistical or otherwise).
<b>15. OTHER COMMENTS</b>	Comments on how the team project is going, successes, etc.
<b>16. COST/TIME ESTIMATE</b>	Track time team spends for 1) meetings, 2) Development, 3) Training, and 4) testing for all team members.

Figure 4, Proof-of-Concept Report Template

## FUNCTION SELECTED FOR THE WEB SERVICES PROOF-OF-CONCEPT

### Change of Address

The Workgroup agreed that the Web Services selected must be able to demonstrate the extent to which Web Services are able to expose data, support interconnectivity, enable transaction processing, enable business process management, incorporate reusability, and address security. After considering six other potential candidate concepts, the Workgroup identified “Address Change” as the leading candidate for the “Proof-of-Concept”. Outlined below is the Workgroup’s selection process.

1. In the Web Services March 28, 2002 meeting the Workgroup discussed what approach to take to evaluate Web Services, potential applications, and the evaluation criteria for selecting the applications to use as a POC.
  - A total of 14 candidate applications were presented of which

number seven suggested a Directory Address Change POC.

- The criteria established for the baseline for selecting development candidates were:
    - a. Interoperability
    - b. Reusability
    - c. Comparability
    - d. Resource Requirements
    - e. Risk
    - f. Potential Business Value
    - g. Applicability
2. The Workgroup next met on April 25, 2002 to discuss which applications to use as POCs. After some discussion it was determined that the teams would all implement Address Change functionality because it is a common application that each agency, institution of higher education, and locality dealt with in their normal course of business.
- Types of address identified within the group are:
    - a.Home/Street,
    - b.Garage,
    - c.Mailing,
    - d.Business,
    - e.Billing,
    - f. Guardian, and;
    - g. Contact.
  - The Address Change concept was simple in focus.
  - The Address Change was sufficiently complex enough to be a good candidate.
  - The Address Change would completely exercise the baseline criteria.

*Please see Appendix D for the workgroup's Web Services project nomination form and the related evaluation script.*

## PROOF-OF-CONCEPT SCHEDULE

Start Date	End Date	Status	Description
3/28/02	3/28/02	Completed	Kick off meeting review draft group charter
4/25/02	8/5/02	Completed	Approve group charter and schedule Select Teams Select Pilot Applications
4/25/02	8/5/02	Completed	Workgroup Pilots
5/30/02	5/30/02	Completed	<ul style="list-style-type: none"> <li>Project Update 1</li> <li>Team Presentations</li> <li>Templates</li> <li>Project Plan</li> </ul>
6/10/02	6/28/02	Completed	<ul style="list-style-type: none"> <li>Develop Test Plans with:                             <ol style="list-style-type: none"> <li>Scenarios</li> <li>Expected Results</li> </ol> </li> </ul>
6/20/02	6/20/02	Completed	Project Update 2
7/1/02	7/1/02	Completed	Submit Test Plan To Tim Bass
7/1/02	7/3/02	Completed	Approve Test Plans
7/26/02	8/16/02	Completed <i>Revised Dates</i>	Integrated testing
7/18/02	7/18/02	Completed	<ul style="list-style-type: none"> <li>Project Update 3</li> <li>Preparing for Group Reports</li> </ul>
8/19/02	8/19/02	Completed <i>Revised Date</i>	Teams Final Reports / Presentations
6/25/02	9/5/02	Completed <i>Revised Dates</i>	Prepare Report for COTS
9/5/02	9/12/02	Completed <i>Revised Dates</i>	Workgroup Review of Draft Report
9/13/02	9/13/02	Completed <i>Revised Date</i>	Submit Final Report to COTS
9/24/02	9/24/02	Completed	COTS Meeting and Web Services Presentation
9/25/02	9/26/02	Completed	COVITS Web Services Break-out Session

Figure 5, Proof-of-Concept Schedule

## WEB SERVICE COMPONENTS FOR CHANGE OF ADDRESS

### Three Types of Change of Address Web Services Components

Three different types of Web Services address components were developed:

- Push address updates to other subscribing agencies.
- Pull address information to satisfy queries.
- Pull address information as a result of a notification (in batch or real-

time).

Other Web Services required to support these processes included authentication, status of results (confirmation/exception), and notification. Business partner published authentication Web Services were not used for the Proof-of-Concept as they are inherently proprietary in nature. Latency and bandwidth issues were not the focus of the Proof-of-Concept.

## UDDI DESIGN AND REQUIREMENTS

### VIPNet UDDI Service

The Virginia Information Providers Network (VIPNet) established a private registry UDDI for the Proof-of-Concept in early June. VIPNet's private registry was based UDDI Version 2.0 running on BEA's WebLogic Server. The hardware used was Sun Ultra II with a Solaris 2.7 operating system. Workgroup teams published to UDDI by both Web Interface and API's for their proof-of-concepts. The UDDI URL is [www.uddi.state.va.us](http://www.uddi.state.va.us)

#### Business/Functional Requirements:

- Private UDDI registry accessible by Virginia state agencies and selected partners via web browser or API interface
- For this Proof-of-Concept, the Web Services Workgroup was required a place to store the information that each of the separate groups need. Under the Web Services definition there exists a component called UDDI, which becomes the directory of the information that needs to be shared. It was the UDDI teams responsibility to make the UDDI component available.

#### Project Equipment Description:

##### Hardware

- Sun Ultra II

##### Operating System

- Solaris 2.7

##### UDDI Server

- BEA WebLogic Server 7.0, provided by BEA

##### Modules

- PERL 5.6.1
- PERL Libraries 5.6.5
- Apache 1.3.24
- SOAP Lite 0.55
- XML Parser 2.31

---

*"Having decentralized computer systems in state government makes it difficult to keep common data accurate in separate state agencies. Web Service applications and a centralized UDDI server will help state agencies become more accurate and in sync with their data and thereby, in the long run, more productive and efficient."*

- Scott E. Fowler, VIPNet  
(Team UDDI)

---

*"The Universal, Discovery, Directory and Integration (UDDI) is still an emerging technology for best practices and wide spread adoption. However, it answers a critical need for defining and*

- Java SDK 1.3.1

#### Acquisition and Installation Activities:

- Hardware for the Proof-of-Concept was already in house.
- Load operating system
- Load supporting software applications and modules
- Load WebLogic supporting modules
- Load WebLogic UDDI server

#### Server Requirements:

- For BEA's WebLogic server to run the team needed to install a number of applications on the existing Sun Ultra II box. Before installing them they needed to get packages from the Internet.
- UDDI Server: BEA WebLogic Server 7.0
- Modules:
  - PERL 5.6.1
  - PERL Libraries 5.6.5
  - Apache 1.3.24
  - SOAP Lite 0.55
  - XML Parser 2.31
  - Java SDK 1.3.1

#### Networking Requirements:

- Established DNS entry [www.uddi.state.va.us](http://www.uddi.state.va.us)
- Firewall issues: Needed to open a translation port. Also opened port 7001 for administrative control.

*discovering published web services. The VIPNet/BEA UDDI repository met the workgroups need for a common standard to publish and lookup Web Services. Using the UDDI, while not trivial, provided a location where each agency could independently publish and discover the other agencies Web Services. Without this UDDI standards-based technology, a much higher level of human interaction would have been required for the agencies to achieve interoperability of their Web Services."*

- Will Howery, BEA Systems, Inc. (Team UDDI)

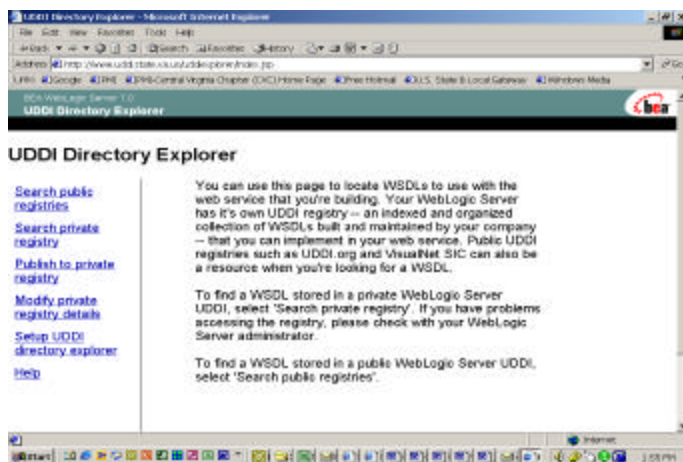


Figure 6, VIPNet UDDI Web site

## Role of UDDI in the Web Services Proof-of-Concept

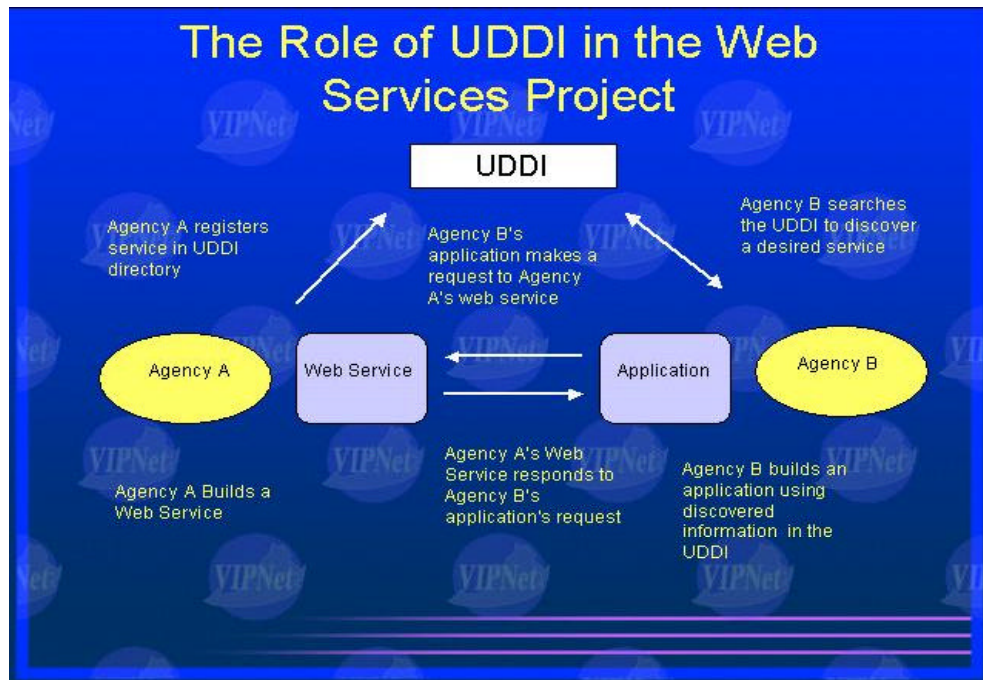
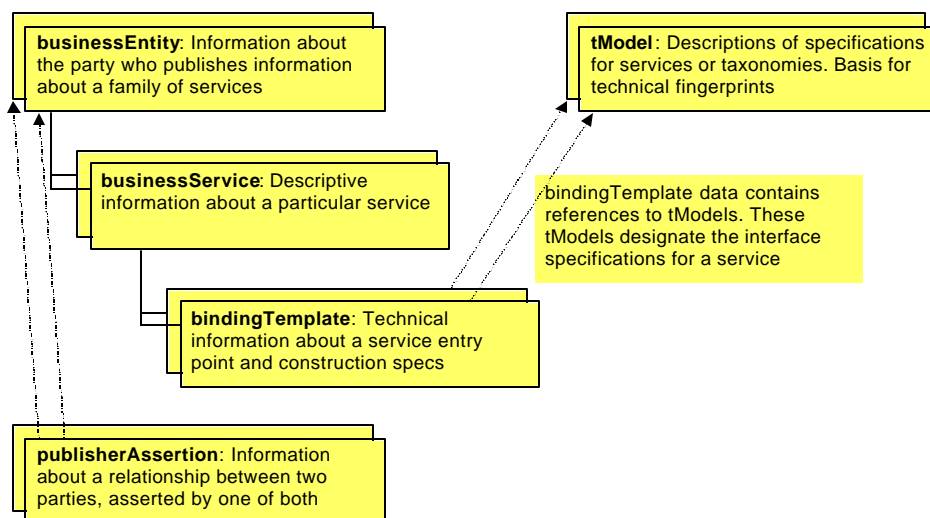


Figure 7, VIPNet Presentation Role of UDDI in Web Services

## UDDI Data Model



Source – From the **UDDI Version 2.0 Data Structure Reference**

Figure 8, UDDI Data Model

**UDDI (Universal Description, Discovery and Integration):** The directory technology used by service registries that allows the directory to be searched for a particular Web service. In essence, UDDI is a White Pages or Yellow Pages that can be used to locate Web Services. There can be both private and public UDDI directories



## Web Interface

- Search Private Directory
- Publish to Private Directory
- Modify Private Registry details
- Setup UDDI directory explorer

## Application Program Interface (API):

- **Publishing**

Enables programs to save and delete each of four data types supported by UDDI. Authenticated access is required to use the Publisher API, i.e. the user must first sign-up with one or more operator sites to establish user credentials.

Save functions:

- save\_business
- save\_service
- save\_binding
- save\_tModel

Delete functions:

- delete\_business
- delete\_service
- delete\_binding
- delete\_tModel

Security functions:

- get\_authToken
- discard\_authToken

- **Searching**

Provide programs with the capability to locate candidate businesses, Web Services and drill into the specifics based on overview information provided in the initial calls. The Inquiry API functions are exposed as SOAP messages over HTTP. No authentication is required to make use of the Inquiry API functions.

Find functions:

- find\_business
- find\_service
- find\_binding
- find\_tModel

Get Details functions

---

### **API (Application Program Interface)**

The interface (calling conventions) by which an application program accesses operating system and other services. An API is defined at source code level and provides a level of abstraction between the application and the kernel (or other privileged utilities) to ensure the portability of the code.

---

### **TModel**

A TModel is the UDDI definition of an Operation in the WSDL

- get\_businessDetail
- get\_serviceDetail
- get\_bindingDetail
- get\_tModelDetail

## DESIGN ISSUES ENCOUNTERED AND RESOLVED

### WSDL Schemas

The Web Services Directory Language (WSDL) presented one of the more challenging design issues. To meet the challenge, a common set of WSDL schemas were established to address parameters, log structures, and return codes etc. to ensure they were deployed in a consistent manner in Web Services, related interfaces, and UDDI setup.

The original workgroup WSDL design prescribed a single service with three operations calling the getAddress, ChangeAddress, and queryAddress. Each operation in the interface either took an AddressData structure as a parameter or returned it as an object. In the query Address operation, the AddressData structure was passed back as an Array. The design of this WSDL was purposefully selected to demonstrate the ability to handle two levels of data complexity. The first level was the AddressData object itself. Returning and converting a complex structure requires the various Web Services servers and clients to be able to convert their native object types to an XML structure and then convert that XML document back into an object structure on the client software. The second level of data complexity demonstrated the ability to take a collection of these complex objects and return them in the Web Services prescribed array XML schema data type.

The Original web service definition prescribed the following structure:

WSDL	Service	Operation
AddressChange	AddressChangeBean	AddressData getAddress(String addressID, String addressType) throws InvalidAddressID;
AddressChange	AddressChangeBean	boolean changeAddress(AddressDat a aAddressData);
AddressChange	AddressChangeBean	AddressData[] queryAddress(Date startDate, Date endDate, String addressType, String OriginatorURL);

Figure 9, Package "A": The original definition

**WSDL (Web Services Description Language):** The language used to create service descriptions of Web Services. It can be used to describe the location of the service and how to run it, as well as what business is hosting the service, the kind of service it is, keywords associated with the service and similar information.

**Array**  
A collection of identically typed data items distinguished by their indices (or "subscripts"). The number of dimensions an array can have depends on the language but is usually unlimited.

For explanation purposes we labeled this original web service definition Package “A”. As the workgroup progressed and the various agencies built their Web Services another WSDL definition was implemented by several of the agencies as follows:

WSDL	Service	Operation
GetAddress	getAddress	AddressData getAddress(String addressID, String addressType)throws InvalidAddressID;
ChangeAddress	changeAddress	boolean changeAddress(Addr essData aAddressData);
QueryAddress	queryAddress	AddressData[] queryAddress(Date startDate, Date endDate, String addressType, String OriginatorURL);

Figure 10, Package “B”: Each operation separated into its own Service and WSDL.

The most significant difference between Package “A” and Package “B” was that Package “A” contained all three Web Services and Package “B” defined each service separately. Both packages were functionally identical, but Package “B” had an impact on the final complexity required to implement clients to utilize the Web Services. The following list details the complexities introduced by Package “B”:

Multiple SOAP ports – need to be defined and configured, and maintained for multiple services. If multiple services are not required their introduction in an integration project adds to complexity without compensating results. Not all teams needed to use the multiple SOAP ports functionality.

Multiple WSDLs – are adding complexity for the client. Again this adds to complexity without compensating results. Best practices encourage designs to use the “Keep it Simple” axiom.

Multiple Static Clients – for a static client, each WSDL must be consumed and produce its native code implementation to invoke and consume that web service. However, if the lookup into the UDDI with a dynamic invocation had been completely implemented the separate WSDLs for each operation would not have added complexity to the clients. But, since the state of Web Services at this time is static clients, a separate WSDL for each operation adds work and complexity

to the clients.

WSDL Interoperability – the workgroup was purposefully setup to have a loosely coupled design process to validate if Web Services would work when the design teams had little interaction. For any integration process to be successful, there needs to be well-known interfaces and definitions setout to prescribe how systems will interoperate. In Web Services the WSDL is that definition. The Workgroup teams achieved interoperability with different agencies following different patterns for packaging the WSDL; this capability presents a strong statement for the viability and flexibility of Web Services.

### Interoperability:

The following table provides the status on each team's interoperability issues.

Feature	VABC	DMV	VRS	GMU	DHRM	DOE/ DMAS	Roanoke/ VA Beach
WSDL Packaging "A"	Y	Y	N	N	Y	X	Y
WSDL Packaging "B"	N	N	Y	Y	N	X	N
WSDL URL Access (Available)	Y	Y	Y	Y	Y	N	N
BEA Client Software could consume the WSDL (in the case of package "B" each operation is listed)	Y	Y	N	Y	Y		N
WSDL Port Access (Available)	Y	Y	N	Y	NT		
Published UDDI Organization with Single UDDI Service	Y	Y	Y	Y	Y	Y	Y
Published Service Properties	Y					Y	
Published to UDDI Programmatically	Y	X	X	X	X	Y	X
Handled Complex AddressData (Service)	Y	Y		Y	NT		
Handled Complex Array Address Data (Service)	Y	Y		N	NT		
Handled Complex Array Address Data (Client)	N				NT		
Test Operations completed getAddress	Y	Y	NT	Y	NT		
Test Operations completed changeAddress	Y	Y	NT	Y	NT		
Test Operations completed queryAddress	Y	Y	N	N	NT		

Figure 11, Team Interoperability

Where:

- Y = Yes
- N = No
- X = Unknown
- NT = Not Tested, have not yet completed testing
- Blank = Indeterminate, a previous dependant feature failed, could not be validated.

One team developed interfaces that successfully demonstrated Web enabling of mainframes (IBM & Unisys) for two agencies. Since for Virginia government the “real world” is comprised of numerous mission critical legacy applications on different platforms, demonstrating mainframe interoperability was essential.

### **UDDI Publishing and Properties:**

The Workgroup's original objective with the UDDI was to have each agency publish their capabilities into a UDDI with properties describing what type of address changes that agency was ready to receive from other agencies. The process was supposed to work as follows:

1. Agency A published an Organization, definition and a Service definition for getAddress, changeAddress, and queryAddress. Each Service would contain properties to identify if that service was subscribing to a certain type of address e.g. Mailing, Business, etc.
2. Agency B upon receiving an address change through its business processes would search the UDDI for a service that had the matching address type in its properties definitions.
3. When Agency B found Agency A's Service definition, Agency B would access Agency A's WSDL URI and dynamically invoke Agency A's service operation to change A's Address

While this loosely coupled process is the eventual promise of Web Services, we encountered several challenges that inhibited its total implementation.

The issues are as follows:

1. Visibility of properties attached to services – In a perfect UDDI world the service publication would follow the WSDL definition and have one service with multiple operations or “bindings” in UDDI terminology. The service would have a binding for the getAddress, ChangeAddress, and queryAddress WSDL URI TModel definitions. However, the browsing tools available do not navigate down into this level of property settings and TModel. If we had used a single UDDI service such as “AddressChange” then the agencies would have had difficulty browsing down into the TModels to identify which Binding defined which address type and WSDL URI. As the project moved forward it became apparent that programmatic access to the low level TModels was not going to happen, this required having our UDDI definitions in a human readable structure.
2. Complexity of the UDDI programming model – The UDDI programming model is very flexible and has a very extensible

structure. This flexibility, however, brings with it a high degree of complexity. The Object model has multiple locations to store “description”, “URI”, and “Properties”. To say the least it is confusing, and best practices are not well defined. This complexity lead to several false starts and restarts by the teams in publishing to the UDDI. At first every agency team had multiple organizations published, and then multiple service etc. In the end every agency did get their Organization, Services, and WSDLs published in the same structural manner. However, most organizations used a manual browser to publish rather than programmatically publishing to the UDDI.

3. Complexity of dynamically invoking WSDL services – The workgroup’s objective was to search the UDDI, retrieve the URI for a service’s WSDL, dynamically consume that WSDL, and dynamically bind to that WSDL. This proved much more complicated for all the business partners than originally thought. As stated earlier, the workgroup purposely introduced the AddressData complex data type to validate the ability to handle “real world” data. The AddressData type became a stumbling block to dynamic invocations. After a considerable effort to make the dynamic invocations work, the workgroup agreed to move back to the more reliable static client invocation.

**Note:** The introduction of the Type “B” WSDL interfaces introduced more difficulty for dynamic invocations. With the Type “A” WSDL interfaces, a service did one search in the UDDI, got one WSDL and with that WSDL made the presumption that they could make multiple calls to getAddress, changeAddress, and queryAddress. With the Type “B” WSDL interface, that presumption (valid by the original WSDL definition) produced errors. The Type “B” WSDL also increased network traffic and processing time to consume multiple WSDL definitions.

4. Availability of UDDI – The original UDDI installed at VIPNet was not interoperable, however a product patch cleared that problem quickly. The final UDDI environment experienced some instability with occasional server locking, causing the need to reboot. This was not a problem with the UDDI specifications, but rather a product implementation issue that requires more research to fully resolve.

## Identification

Defining/establishing a Universal Identification Number (UID), i.e., customer number, was resolved using a 9 numeric character field, and generating the UID for test data using a sequential number generator.

## Administration

Identifying who has administrator functionality was resolved by assigning

---

### URI (Universal Resource Identifier)

Internet space is inhabited by many points of content. A URI is the way to identify any of those points of content, whether it is a page of text, a video or sound clip, a still or animated image, or a program. The most common form of URI is the Web page address, which is a particular form or subset of URI called a Uniform Resource Locator (URL). A URI typically describes:

- The mechanism used to access the resource
- The specific computer that the resource is housed in
- The specific name of the resource (a file name) on the computer

one person from each team as the administrator with a password/PIN.

### **Loop Protection**

An address change triggers a “propagation” of that change to other change of address Proof-of-Concepts only when made by the originator of the change.

### **Locality Check**

Concerning whether all address changes should be sent to localities, or only their jurisdictional address changes, for this Proof-of-Concept all relevant types of address changes were sent to Roanoke and Virginia Beach, regardless of address location.

### **Entity Type Check**

In order to establish a means to distinguish between an individual and a business entity, it was determined that exercising this business requirement did not add value to the interoperability of Web Services, although in the real world there may be a business requirement to distinguish between these two entity types.

### **Batch vs. Real Time Processing**

In order to state when “pull” is initiated after a “notification” it was determined that following a “notification”, whether an address change “pull” was initiated by an application in “real-time” or at a later “batch” window did not add value to the interoperability of Web Services. Rather, this is a business decision, which would address performance related issues and risks (e.g., latency and band-width).

## **TEST DATA**

The workgroup established test data and test use cases that yielded satisfactory unit test results for all Web Services components and functions as well as the UDDI. For greater test case details, please see Appendix sections D through K.

## **SECURITY FOR FIRST-GENERATION WEB SERVICES**

For the Commonwealth of Virginia to take advantage of the promise of Web Services, there is a need to be able to feel comfortable with the security of the data exchange. Web Services can possibly be seen as having three different scenarios from a security perspective. The first scenario is its use as a common interface to exchange data between disparate, yet known partners. The second scenario involves its use between unknown users. A third scenario, which is a variation of both one and two, is also presented to illustrate the complexity of the issues involved.

---

### **OSI (Open System Interconnection)**

OSI (Open Systems Interconnection) is a standard description or “reference model” for how messages should be transmitted between any two points in a telecommunication

## Background

When discussing security in the web arena, one usually refers the three A's (AAA) of security. They are: Authentication, Authorization/Access Control, and Auditing.

Authentication is "who you are" and can run the spectrum from anonymous to strong. It is used for purposes such as login to a site but can also include non-repudiation elements in documents or transactions. Anonymous is what is usually in effect when browsing a public web site. The site is, by definition, public. Strong authentication requires a fairly high level of certainty that you are who you say you are. This can be via two-factor authentication (what you know and what you have) such as a token-based one-time password or it can be via certificates, source IP addresses in TCP connections, and passwords etc. This category also should include data integrity and/or non-repudiation since it refers to the authenticity of the data.

Authorization is the "what you are allowed to do." This would include: access to a site, a part of a site, a particular record/file from the site. It can be "read-only," "read-write." It defines access. Encryption is included with the Authorization category since it is used as a tool to enforce authorization.

Auditing provides the forensics of a system. Can it be logged who did what when? How granular is it? Granularity is a good and bad thing. Too much data and it is easy to miss abnormalities. Too little data and the abnormalities may not show up or may not have enough to make a good forensic case if needed. For the most part, it is assumed that it can either be deployed or at least build sufficient auditing capabilities.

One final caveat, there must always be an examination of what processes or data are being protected. An analysis of the security requirements for that specific process must be performed. An appropriate level of security should be applied based on these requirements.

### **Scenario One: Known partners**

In this scenario, it is known who is (or should be) on each end of the transaction. There may still be a wish to publish services via WDSL to a UDDI so that change can be managed. However, it is still known who is on the other end.

By using today's web toolbox to secure these transactions, this scenario is the easiest to implement. The whole spectrum of authentication and authorization tools can be found here as discussed above.

network. Its purpose is to guide product implementers so that their products will consistently work with other products. The reference model defines seven layers of functions that take place at each end of a communication. Although OSI is not always strictly adhered to in terms of keeping related functions together in a well-defined layer, many if not most products involved in telecommunication make an attempt to describe themselves in relation to the OSI model. It is also valuable as a single reference view of communication that furnishes everyone a common ground for education and discussion.

**SAML (Security Assertion Markup Language):** An Extensible Markup Language (XML) standard that allows a user to log on once for affiliated but separate Web sites. SAML is designed for business-to-business (B2B) and business-to-consumer (B2C) transactions. SAML specifies three components: assertions, protocol, and binding. There are three assertions: authentication, attribute, and authorization. Authentication assertion validates the user's identity. Attribute assertion contains specific information about the user. And authorization assertion identifies what the user is authorized to do.



### **Scenario Two: Unknown visitor**

Currently, this scenario is the toughest to handle when any of the stronger levels of authentication or authorization are needed because there is no pre-established arrangement that can be used to help.

### **Scenario Three: Multi hop SOAP messaging**

This scenario further complicates both scenario 1 & 2. One of the promises of SOAP is to take one message and pass it through multiple web services, each web service completing its part of the total request. This implies that there may be varying requirements for authentication (who started this and how do we know that) and authorization (varying requirements for encryption within the message and the requirement that some processors may not be allowed to see other parts of the message).

### **Discussion**

There are several areas where Web Services can be implemented. However, the true promise of Web Services cannot be fully realized until new standards are fully developed and adopted. Several, sometimes competing, standards are currently being considered for various aspects of the security puzzle as follows:

- XKMS: A key management specification that describes the distribution and registration of public keys. Developed by W3C.
- XML Encryption: Encryption of XML content. This protocol, among other things, addresses encryption of sections of the SOAP message (scenario three). Developed by W3C.
- XML Signature: Signing (non-repudiation) of XML content. Developed by W3C.
- WS-Security: Microsoft's tool on using XML Encryption and XML Signature.
- Liberty Alliance: Sun Microsystems, Inc. backed solution for authentication.
- ebXML: Electronic Business XML. Jointly sponsored by Oasis and UN/CEFACT, this is a suite of specifications being designed to facilitate ebusiness. ebXML can also be defined as an XML version of EDI.

Web Services security specifications are still under development, therefore a common security mechanism for team-to-team Web Services processing was not considered part of this Proof-of-Concept. However, scenario one options were certainly available.

## TRAINING

The team received varying degrees of formal training from their business partners as well as on-the-job, self-learning and consultation with experienced developers. The teams trained for 250 hours overall, an average of 42 hours each.

On the whole, the training consisted of a general overview of Web Services concepts and the business partner's software products, reading related white papers, conference calls with developers and other teams, and one-on-one consultation with the business partner's developers. The team that utilized free software immediately experienced the need for consultation and guidance from knowledgeable trainers.

In most cases training was structured as follows:

- Self Study – XML Concepts
- Self Study – SOAP Concepts
- Self Study – UDDI Concepts
- Self Study – WSDL Concepts
- Business Partner's Product
  - Overview of architecture including software features and limitations
  - Drilldown on each of the product's components (most included a developer workbench)
- Training/Development/Testing Sessions

### **Individual team training requirements are outlined below:**

**Team 1** – Department of Motor Vehicles (DMV), Microsoft Corporation, Susquehanna Technologies

Training Requirements:

- DMV representatives were sent to the .NET training provided by Microsoft at DIT on June 6 & 7.
- No other training is required for this project.

**Team 2** – Department of Alcohol and Beverage Control (ABC), BEA Systems, Inc.

Training Requirements:

- Self Study – XML
- Self Study – SOAP

- Self Study – UDDI
- Self Study – JXM, JXRPC
- Self Study – WSDL
- BEA assisted – Weblogic / Web Services Integration
- BEA assisted – Workshop
- BEA assisted – technical guidance

**Team 3** – Department of Human Resources Management (DHRM), Virginia Retirement System (VRS), SilverStream, Inc.

Training Requirements:

- SilverStream will be dependent on both agencies personnel for infrastructure support, and Mitem support at DHRM.
- Training on SilverStream products: Application server, Composer Server, Composer Designer, Workbench IDE, and 3270 Connector (VRS)
- Training on the following concepts: Java, HTML, SOAP, WSDL, Web Services, XML, and XSL

**Team 4** – County of Roanoke, City of Virginia Beach, Sun Microsystems, Inc.

Training Requirements:

- Introduction to Web Services White paper
- Sun One Studio (formerly Forte) Tutorial
- One on one conference calls

**Team 5** – Department of Medical Assistance Services (DMAS), Department of Education (DOE), Software AG, Inc.

Training Requirements:

- The team received two hours high level training June 12, 2002, covering the following points:
  1. General overview of Web Services
  2. General overview of Software AG's products
    - Each developer also received a book titled XML and Web Services, Unleashed SAMS.
    - Each team member will track time spend studying/training for our Proof-of-Concept.
  3. June 26, 2002 Software AG provided DOE and DMAS technical staff with overview of architecture and drilled down on each of the components.

4. July 24, 2002 Software AG, DOE and DMAS technical staff conducted a 3-hour, training/development/testing session. Software AG provided a review of all application software and discussed various features and limitations of the product at that time. The underlying concepts of Web Services were re-capped in light of discussion at the July 17th workgroup meeting.
5. The majority of the training required to implement the POC has been in the form of on-the-job self-learning and consultation with experienced developers.

**Team 6** – George Mason University (GMU), WebMethods, Inc.

Training Requirements:

- Although a developer training class was repeatedly offered by webMethods, scheduling conflicts prohibited attendance. However, lack of training was not a factor in the completion of the Proof-of-Concept project.

**Team UDDI** – Virginia Information Providers Network (VIPNet), BEA Systems, Inc.

Training Requirements:

- Overall there wasn't any training outside of individual research and consultations from business partners. All of the information that we used was found via the Internet, through books purchased, and business partner consultations.
  1. Purchased O'Reilly books on Web Services
  2. Consultation from BEA on UDDI system
  3. Consultation from Software AG

A general consensus developed among the workgroup that buying support/consultation and receiving product training were essential to the successful implementation of a Web Services project. For more detailed information on training please refer to the Level of Effort in section IV or the individual team reports in the Appendix.

## BEST PRACTICES

### LESSONS LEARNED FROM INDUSTRY

A recent Gartner article discussed overarching Web Services lessons<sup>v</sup>. It suggested that one of the lessons learned by enterprises involved in initiatives that use visionary or generally experimental technology is that setbacks are common. Enterprises also reported that they had deployed Web Services in projects with low developer headcounts. Typical internal project teams are closer to three developers (which tracks with the workgroup's average); external development is generally conducted in such a way that the load for enterprises is distributed to teams of similar size (for instance, two internal and one external developer).

### BEST PRACTICES DERIVED FROM THE PROJECT

The following are fundamental best practices derived by the Workgroup:

- Web Services are technically ready for implementation. As with any new technology, maturity levels could impact some areas of development. But overall, the technology and the paradigm work.
- The tools are maturing in terms of Web Services development but are not quite there for testing.
- Interoperability requires significant coordination. One of the major hindrances to interoperability was the lack of detailed specifications.
- Control the web service environment. Seek to implement Web Services for lower risk, less complex business processes.
- UDDI and subscription services must be implemented early in Web Services development process and be carefully controlled.
- UDDI as a directory service for Web Services is flexible but complex. Grasping a complete understanding of WSDL and UDDI can be difficult. Unless public directories are a definite requirement, we would recommend taking the approach the workgroup ended with – to statically bind the services.
- When converting a WSDL created from another development platform, manually verify that the WSDL generation was correct.

---

*“From internal experimentation to inter-enterprise business-to-business linkage, Web Services are filtering into daily IT practice. 2003 is the year for even cautious enterprises to begin Web Services pilots.”*

Whit Andrews, GartnerGroup<sup>vi</sup>

- Naming of parameters in the WSDL should not use common terms that may be reserved words. For example “string” is commonly used as a data type.
- Agree upon a WSDL and strictly adhere to it in implementing Web Services to avoid the potential of unnecessarily increasing the complexity of attaching Web Services.
- Keep the solution as simple as possible while still meeting the design requirements. This will help speed up the process of development and guarantee success.
- Robust exception handling should include the judicious use of SOAP exceptions. This would allow the calling process to receive more detailed information in the event of a system error.
- When creating a set of web service components, much of the upfront design time should be spent in determining the optimal programming interface to the components. Once implemented, the interface is very difficult to change without affecting subscribers to the web service.
- XML structures (list information) must be identified.
- When passing XML-based data to a web service, the originating process should validate the data by comparing it to the appropriate XML schema or DTD.
- The XML going in or coming out of a Web Services should be verified against a schema.
- Change management processes should be utilized in order to avoid confusion between web service interfaces and function/data requirements.
- A common set of data for interoperability testing and web service cross communication is critical and will need to be carefully coordinated between participating organizations.
- Unique identifiers must be defined for companies and individuals.
- Common data validation rules should be implemented, if at all possible, so data errors can be minimized/avoided when calling the provided services.

- Standard, customary software practices should be adhered to when developing Web Services, especially in the areas of design.
- The Web Service, that is being registered, should be internally consumed before publishing the specifications to the UDDI server.
- If free software is utilized:
  - Need to have access to experienced resources,
  - Need to provide the necessary training to staff, and;
  - Consider buying support from reseller.

## TOOLS AND RESOURCES

### TOOLS

**Team 1** – Department of Motor Vehicles (DMV), Microsoft Corporation, Susquehanna Technologies

#### Project Equipment Description:

Hardware – provided by Compaq

- 2 Compaq Proliant Servers ML Series
- (2) 72 GB Hard Drives
- 512 MB Memory
- Internal tape
- 2 – 17” monitors
- 1 Workstation for application development and testing of remote access to Web Services

Software

- Visual Studio.NET – Web Services Development
- All components for XML, WSDL, SOAP, UDDI
- Windows 2000 Advanced Server for hosting platform. Also included in the installation is the .NET Framework to provide Web Services capabilities
- SQL Server 2000 for database storage and analytic processing of data
- Microsoft provided the software

Location

- The servers are housed at VIPNet.

Acquisition and Installation Activities

- Microsoft acquired equipment for the purposes of the pilot project.

Server Requirements

- There are no additional requirements above and beyond the equipment described above.

Networking Requirements

There are no specific networking requirements other than the



following:

- Both Compaq servers must be on the same network segment.
- The web server must be available on the Internet allowing connections via the HTTP protocol and ports 80 and 443 (inbound and outbound).

**Team 2** – Department of Alcohol and Beverage Control (ABC), BEA Systems, Inc.

Project Equipment Description:

Hardware

- Dell Laptop
- 1 GHZ Intel Processor
- 30 GB Hard Drive
- 512 MB Memory

Software

- Windows 2000
- BEA Weblogic Application Server
- BEA Weblogic Workshop
- Pointbase Database

Acquisition and Installation Activities

- Additional hardware will not be required for this prototype. BEA has agreed to supply copies of the Weblogic application server and Workshop tool.

Server Requirements

- The purchase of additional resources is not required.

Networking Requirements

- The server and entire application environment will run outside of the VABC firewall. This will require the system administrator to setup a unique IP address for this machine. No additional networking requirements are anticipated.

**Team 3** – Department of Human Resources Management (DHRM), Virginia Retirement System (VRS), SilverStream, Inc.

Project Equipment Description:

Hardware

- 1 Windows NT Workstation/Server 4.0 or Windows 2000 provided by VRS
- 1 Windows NT Workstation/Server 4.0 or Windows 2000

---

**Hypertext Transport Protocol (HTTP)**

The set of rules for exchanging files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web

provided by DHRM

- Both machines must have, at a minimum, 512 MB RAM and 260 MB disk space
- Both machines will be given external access so that other agencies can access these machines.

#### Software

- 1 Not-for-resale SilverStream eXtend Application Server 3.7.4
- 1 Not-for-resale SilverStream eXtend Composer Enterprise 3.5
- 1 Not-for-resale SilverStream eXtend Composer Developer 3.5
- 1 Not-for-resale SilverStream eXtend Composer 3270 Connector (for VRS)

#### Acquisition and Installation Activities

- To support marshalling of complex types in a Web Service, both agencies were upgraded to the latest and greatest version of SilverStream products: SilverStream 3.7.5, Composer 4.0, Workbench 4.0, and Jbroker Web 2.0 have been installed on server machine.

#### Server Requirements

- None

#### Networking Requirements

- None

**Team 4** – County of Roanoke, City of Virginia Beach, Sun Microsystems, Inc.

#### Project Equipment Description:

##### Hardware

- Sun Solaris Enterprise 220R server with two 450MHz
- UltraSPARC-II processor, 4MB E-cache, 1GB memory, two 36GB disks

##### Software

- Sun One Web Server
- Sun Ine Application Server
- Sun One Studio (Java IDE)
- Sun One Directory Server
- Pointbase Database (freeware)

#### Acquisition and Installation Activities

- None

#### Server Requirements

- Original plan was for servers to be installed in VA Beach and Roanoke. The first server was installed in Roanoke and later removed for repairs. The plan was later revised, because of time constraints, to have one server installed at the consultant's office.

#### Networking Requirements

- Re-configured to allow access to the server where the Web Services reside. Ports were opened to access application server, database, and web server.

**Team 5** – Department of Medical Assistance Services (DMAS),  
Department of Education (DOE), Software AG, Inc.

#### Project Equipment Description:

##### Hardware

- Sun Solaris

##### Software

- Apache HTTP Server 1.3
- Apache Jakarta Tomcat Servlet & JSP Engine 3.3
- Apache SOAP Implementation 2.3
- Software AG Tamino XML Server (including X-Node ODBC) 3.1
- Software AG EntireX XML Mediator 7.1
- Agency Representative Software: Oracle RDBMS 8.1

##### Location

- Hardware and software is hosted by Software AG, Inc. in Reston, Virginia

#### Acquisition and Installation Activities

- Additional software will be acquired such as freeware development tools and will be listed here as obtained.

#### Server Requirements

- Configured Apache Server for HTTPS and SSL. However, secure communications were never utilized.

#### Networking Requirements

- Re-configured firewall to allow access to the Tomcat Server where the Web Services resides. Due to Software AG's network security policies, we had to clarify the use of the server for IT and compromise on server availability. We

opened all necessary ports, but made the box available only during the hours of 7AM and 7 PM.

**Team 6** – George Mason University (GMU), webMethods, Inc.

#### Project Equipment Description:

##### Hardware

Hardware Provided by GMU and located at GMU

- Gateway Pentium IV
- Ghz 4 CPU
- 512 MB RAM/40 GB Storage
- Windows XP Professional
- Sun E-420R Ultra-80
- 4/450 Mhz CPUs
- 4 GB RAM/1 TB Storage
- Solaris 2.8

##### Software

Software (Provided by webMethods and installed at GMU)

- webMethods Integration Server
- webMethods Developer
- webMethods Enterprise Server

Software (provided by GMU and installed at GMU)

- Oracle 8i

##### Acquisition and Installation Activities

- webMethods Integration Server, Developer, Business Process Modeling, and Workflow software acquired and installed (no cost to agency). Updated trial license keys on 6/18/02 and 7/16/02

##### Server Requirements

- No server changes or acquisitions required

##### Networking Requirements

- Static IP address obtained and DNS entries established on 7/16/02

## RESOURCES

**Team 1** – Department of Motor Vehicles (DMV), Microsoft Corporation, Susquehanna Technologies

**Team 2** – Department of Alcohol and Beverage Control (ABC), BEA Systems, Inc.

**Team 3** – Department of Human Resources Management (DHRM), Virginia Retirement System (VRS), SilverStream, Inc.

**Team 4** – County of Roanoke, City of Virginia Beach, Sun Microsystems, Inc.

**Team 5** – Department of Medical Assistance Services (DMAS), Department of Education (DOE), Software AG, Inc.

**Team 6** – George Mason University (GMU), WebMethods, Inc.

**Team UDDI** – Virginia Information Providers Network (VIPNet), BEA Systems, Inc.

## LEVEL OF EFFORT

### Personnel and Time

The workgroup comprised of six change-of-address Proof-of-Concept teams in addition to a UDDI team. Each team was consisted of public and private sector personnel. Seven state agencies, one institution of higher education, one county and city, and six private-sector software providers participated in the project (several other private-sector software integrators were added as the project progressed). Team size ranged from four to 13 individuals. Many team members were geographically spread out across the commonwealth. In most cases the leader was a senior public sector IT manager supported by senior private sector personnel as well as other public sector associates. The workgroup spent seven months and a total of 2,480 hours on the project, which averaged out to 354 hours per team.

---

*“Web presence is not optional for governments in the United States. Citizens are online and learning to demand answers at Internet speed. Government budget-writers require that the cost-savings potential of the Internet be mastered.”*

Pew Internet & American Life Project<sup>vii</sup>

Outlined below the individual team personnel and time figures:

Team Name	Personnel	HOURS					Total Hours
		Meetings	Development	Training	Testing	Documentation	
UDDI	5	28	49	30	18	30	155
Team 1 – DMV & Microsoft	11	31.5	174	16	32		254
Team 2 – ABC & BEA *	4	6	60	26	50		142
Team 3 – DHRM, VRS & SilverStream	10	60	640	80	20		800
Team 4 – Va Bch, Roanoke Cnty & Sun	13	100	120	80	20		320
Team 5 – DMAS, DOE & Software AG	10	158	311	34	41	25	569
Team 6 – GMU & webMethods	5	64	112		64		240
<i>Overall Total Personnel/Hours Spent on Project</i>	<i>58</i>	<i>448</i>	<i>1466</i>	<i>266</i>	<i>245</i>	<i>55</i>	<i>2,480</i>
<i>Average Personnel/Hours Spent on Project</i>	<i>8</i>	<i>64</i>	<i>209</i>	<i>44</i>	<i>35</i>	<i>28</i>	<i>354</i>

## APPENDICES

Appendix A: Web Services Charter for Proof-of-Concept Initiative.....	56
Appendix B: Web Services Proof-of-Concept Schedule.....	59
Appendix C: Virginia Technology Achievements .....	61
Appendix D: Web Services Project Nomination Form & Evaluation Script	71
Appendix E: Glossary of Terms .....	73
Appendix F: Team 1 Final Report .....	81
Appendix G: Team 2 Final Report .....	105
Appendix H: Team 3 Final Report .....	127
Appendix I: Team 4 Final Report .....	141
Appendix J: Team 5 Final Report .....	147
Appendix K: Team 6 Final Report .....	162
Appendix L: UDDI Team Final Report .....	167
Appendix M: Web Services Resources and Business partners .....	175
Appendix N: Web Services Address XML Schema.....	177

## Appendix A: Web Services Charter for Proof-of-Concept



### COTS Web Services Pilot Project Initiative Workgroup Charter

**Revised May 30,2002**  
**COTS Approved July 11,2002**

#### GENERAL INFORMATION

<b>Workgroup Name:</b>	Web Services Pilot Project	<b>Date Established:</b>	11/08/01
----------------------------	-------------------------------	--------------------------	----------

#### WORKGROUP MISSION

- **Introduction**

"Web Services" technology is seen as a potential next-step in the evolution of application development in support of real-time business transactions. Technically, Web Services represents the creation of platform-neutral web-based applications (or "services") that can be shared and/or re-used by any entity capable of communicating using the proper protocols. The value appears to be in how the services are loosely coupled to facilitate a broad range of very simple or complex processes that may also involve integration with existing legacy systems. Greater sharing and reusability amongst applications, with platform neutrality, is an area deemed worth exploring given the significant investment in web technology (in general) within the Commonwealth.

The following Internet language and/or protocol specifications currently comprise Web Services: Extensible Markup Language (XML), Simple Object Access Protocol (SOAP) and Web Services Definition Language (WSDL). Please reference the [www.w3c.org](http://www.w3c.org) web site for more information on their technical definitions. In addition, other protocols and specifications such as Universal Description Discovery Integration (UDDI) and Lightweight Directory Access Protocol (LDAP) play a role in the evolving Web Services definition and, in fact, may be incorporated into



business Partner products marketed as providing support for the development and implementation of Web Services.

It should be clearly noted that this project involves R&D activities as they relate to Web Services (and supporting) technologies. The implementation of any production system as a result of these activities should not be assumed.

- **Vision**

Within the established schedule, the objective of this workgroup is to have evaluated Web Services and supporting technologies that promote the development of sharable applications accessible via the Internet and/or organizational Intranets.

- **Mission**

Determine the viability of Web Services as an application development and integration approach that will create a secure, standards-based, platform independent framework for interoperability that supports activities of the Commonwealth of Virginia.

## WORKGROUP GOALS AND OBJECTIVES

The Web Services pilot project workgroup will support COTS strategic goals. The alignment of COTS strategic goals and workgroup objectives is shown below. The workgroup's work plan developed as a result of this charter will, at a minimum, establish measures of success for each objective and provide for regular workgroup performance reporting to COTS.

<b><i>COTS Strategic Goals</i></b>	<b><i>Workgroup Objectives</i></b>
To assist the Secretary of Technology to monitor trends and advances in fundamental technologies of interest and importance to the economy of the commonwealth.	<p>To determine the probable impact of Web Services technology on the strategic direction of application development and integration in the private and public sectors.</p> <p>Through a technical R&amp;D process, determine the feasibility, advantages, disadvantages and best practices when implementing Web Services in the commonwealth.</p> <p>To determine requirements for implementation that will include (but will not be limited to) hardware, software, training, and security and migration issues. Also, to determine potential cost scenarios.</p> <p>To make a recommendation regarding adoption of Web Services technology for the commonwealth.</p>

## WORKGROUP SCOPE

The scope of the Web Services Pilot Project collaborative research effort is as follows:

- 1) Entity Eligibility – Any state agency, state institution of higher learning, or local government in the Commonwealth may initiate a request to participate in the Web Services Pilot Project. A set of criteria for participant selection will be defined and used to form the team.
- 2) Business Partner Services – Business Partners participating in the Web Services Pilot Project will be asked to provide certain services during the length of the pilot, at no cost to the Commonwealth or any participants. These services include but are not limited to:
  - a) Web Services developer kits.
  - b) Technical/product training.
  - c) Help Desk (product) support.
  - d) Any proprietary Web Services marketed as standard components by the business Partner (for test purposes only).
  - e) Licensing extensions to participants for any required software, middleware, O/S or other product needed to support the pilot test environment.
  - f) Access to Web Services compatible security tools offered by the business partner.
- 3) Duration – The Web Services Pilot Project testing and evaluation will be scheduled to be completed in time to be factored into the input of the Fall 2002 budget as needed. Therefore, with an anticipated start date of February 2002, the pilot is expected to be completed (i.e. final report issued to COTS) by the end of September 2002.
- 4) Systems and Telecommunications – All participants must be willing to make available supporting hardware/software that will enable research and development.
- 5) Security – Security requirements will be identified and tested. Business partners participating in the Web Services Pilot Project will be asked to demonstrate their Web Services compatible security tools.

## WORKGROUP AUTHORITY AND RESOURCES

- **Authorization**

This workgroup charter has been initiated by COTS. Mr. J. Timothy Bass, COTS representative and Chief Technology Officer of the Virginia Retirement System, has been appointed chair of the Web Services Pilot Project Workgroup by the Secretary of Technology.

- **Workgroup Organization**

The following organizational model will be used to facilitate the Web Services Pilot Project:

1. The COTS Web Services Pilot Project ad hoc workgroup is established as a steering committee for the Web Services Pilot Project. This workgroup is chaired by a COTS member appointed by the Secretary of Technology, and it will be composed of the following:
  - a) The workgroup Chair (a COTS member).
  - b) One individual from each Commonwealth entity participating in the pilot.
  - c) A COTS-Enterprise Architecture Workgroup member.
  - d) A DIT representative.
  - e) A DTP representative.

In addition, DTP will provide staff support to the steering committee as necessary.

2. The Web Services Pilot Project Workgroup is responsible for:
  - a) Defining the necessary criteria for participation in the pilot for both Commonwealth entities and business partners.
  - b) Identifying, evaluating and selecting business partners and Commonwealth entities for participation in the Web Service Pilot Project.
  - c) Providing monthly updates and a final report with recommendations to COTS.
3. Each participating entity will provide staff from their organization to conduct the appropriate development, testing and evaluation activities as established in the project plan.

## Major Milestones and Deliverables

To be determined by the workgroup and submitted to COTS for approval at a later date.

## Appendix B: Web Services Proof-of-Concept Schedule

Start Date	End Date	Status	Description
3/28/02	3/28/02	Completed	Kick off meeting review draft group charter
4/25/02	8/5/02	Completed	Approve group charter and schedule Select Teams Select Pilot Applications
4/25/02	8/5/02	Completed	Workgroup Pilots
5/30/02	5/30/02	Completed	<ul style="list-style-type: none"> <li>➤ Project Update 1</li> <li>➤ Team Presentations</li> <li>➤ Templates</li> <li>➤ Project Plan</li> </ul>
6/10/02	6/28/02	Completed	<ul style="list-style-type: none"> <li>➤ Develop Test Plans with: <ul style="list-style-type: none"> <li>➤ Scenarios</li> <li>➤ Expected Results</li> </ul> </li> </ul>
6/20/02	6/20/02	Completed	➤ Project Update 2
7/1/02	7/1/02	Completed	➤ Submit Test Plan To Tim Bass
7/1/02	7/3/02	Completed	➤ Approve Test Plans
7/26/02	8/16/02	Completed	➤ Integrated testing
7/18/02	7/18/02	Completed	<ul style="list-style-type: none"> <li>➤ Project Update 3</li> <li>➤ Preparing for Group Reports</li> </ul>
8/19/02	8/19/02	Completed <i>Revised Date</i>	➤ Teams Final Reports / Presentations
6/25/02	9/5/02	Completed <i>Revised Date</i>	Prepare Draft Report for COTS
9/5/02	9/12/02	Completed <i>Revised Date</i>	Workgroup Review of Draft Report
9/13/02	9/13/02	Completed <i>Revised Date</i>	Submit Final Report to COTS
9/24/02	9/24/02		COTS Meeting and Web Services Presentation
9/25/02	9/26/02		COVITS Web Services Break-out Session

## Appendix C: Virginia Technology Achievements

### Virginia Government

Virginia enjoys an international reputation as a leader in technology. It was the first state to create a cabinet level chief information officer with the establishment in 1998 of the Office of the Secretary of Technology. Virginia, which led the way in 1995 as one of the first states to have a portal on the Internet, has continued to offer and improve on web-enabled government services. This combination of sound management and innovative application helped Virginia net the highest grade from the Government Performance Project, which rates the 50 states and 15 federal agencies on government management, including information technology.

The Council on Technology Services was formed in August 1998 to draw best practices and innovations from some of the best minds in state government. Specific areas covered by workgroups include: Enterprise Architecture, Privacy, Security & Access, and State and Local Network Integration. The Council, chaired by the Secretary of Technology, has 26 members, including representatives from state agencies, institutions of higher education and local governments, and six ex officio members.

1. Virginia was first in the nation to offer drivers the ability to renew their licenses through the Department of Motor Vehicles (DMV) website. Customers conduct the renewal using a computer with a modem, a DMV personal identification number (PIN) and a major credit card. Their new driver's license will be mailed within three business days.
2. Virginia is a leading participant at the state and county levels in the General Service Administration's (GSA) Office of Intergovernmental Solutions (OIS) Government Without Boundaries (GWOB) Teamviii that developed a pilot Web Service offering recreational data.

On May 23, 2002, GSA's OIS held a GWOB Team meeting to discuss the status of cycle I products of the intergovernmental initiative. Participants at the meeting included representatives from the States of Maryland, New Jersey and Virginia, representatives from Fairfax County Virginia, a representative from National Association of State Chief Information Officers, representatives from the Department of Interior (DOI) and of course members from GSA's OIS.

The highlights of the meeting included DOI's demonstration of an enhanced Recreation.gov application (GWOB Cycle I, Parks and Recreation demonstration project) that successfully included the Commonwealth of Virginia's Parks facilities data and the County of Fairfax's Parks facilities data in Recreation.gov facilities search routine. Keith Stewart (DOI) described the two methods used for uploading the state and local data into Recreation.gov. The Fairfax County Parks data was uploaded dynamically using the GWOB XML schema for Parks data. The Commonwealth of Virginia's Parks information was uploaded by Keith from an excel spreadsheet formatted and populated with data according the GWOB Parks schema. This integration of information and services from multiple levels of government is an example of the emerging seamless citizen-centric government of the future.

According to Frank McDonough, Deputy Associate Administrator, OIS, GSA, "the Government

Without Boundaries program has become a model for intergovernmental project management in support of the President's E-government agenda."

3. Governor Mark R. Warner<sup>ix</sup> recently announced that Virginia is the only state in the nation to move up in a national ranking of high-tech job markets. Virginia's high-technology industry added 4,300 new tech jobs last year, for a total of 228,900 jobs in 2001. Virginia has also been notified that it placed in the top ten states in Electronic Commerce, one of three categories in Phase II of the Digital States Survey. Virginia's exact ranking will be made public in August or September.

4. The Governor<sup>x</sup> also publicized that Virginia had been awarded a first-place ranking by the Center for Digital Government for the social services category in the prestigious 2002 Digital State Survey. Virginia earned this honor by implementing industry-standard best practices and overhauling internal structures and technologies in agencies of the Health and Human Resources secretariat.

5. On August 1, 2002, Virginia's electronic procurement system, know as *eVa*, was named one of the five "Best of Breed" technology projects for the year by the Center for Digital Government. Out of about 1,500 submissions, *eVa* was selected in the area of Business and Commerce.

*eVA* was set in motion on October 30, 2000, when a Department of General Services (DGS) contracting team with representatives from the Secretary of Technology, Secretary of Finance, agencies, universities, and local government competitively awarded a contract to American Management Systems (AMS). The contract required AMS to provide electronic procurement services to the Commonwealth with core services available to agencies by March 2001 and full end-to-end procurement available by December 2001.

On March 5, 2001, DGS implemented *eVA* Core Services early adopters program with six agencies, 600 users, 400 registered business partners, and 70 catalogs. At the end of March 2002 *eVA* has 228 agencies, institutions of higher education and local governments online, over 4,260 users, 4,780 registered business partners, and 435 catalogs equivalent to 1.9 million products. Over 15,730 orders representing over 80 million in sales have been processed through *eVA* to date. Within the AMS Ariba electronic marketplace that includes commercial customers, *eVA* is one of the largest customers, with more products and suppliers on line than ChevronTexaco Corporation. By the end of July 2002 the Enhanced Services will be implemented, and by mid-to-late 2002 the Advanced Services will be implemented. A July 2001 survey conducted by Forrester Research entitled, "*States' e Procurement Road Map*," placed Virginia as one of the leading states in this area of technology."

Additional information on Virginia technology achievements in government may be found on the Secretary of Technologies Web site,  
<http://www.technology.state.va.us/TechVA/techinvagov.cfm>.

## Virginia Citizens

1. The Virginia Employment Commission ([www.vec.state.va.us](http://www.vec.state.va.us)) provides numerous employment services to citizens of the Commonwealth. These include job referral and placement, referral to training and job search skill building activities. VEC staff assists employers by screening and referring applicants for job vacancies, providing critical labor market information for business and economic planning and coordinating Employer Advisory Committee activities across the state. Employers placed over 154,490 job openings and the VEC made 497,423 referrals during the program year ending June 30, 2001.

Virginia's Automated Labor Exchange (ALEX) (<http://www.alex.vec.state.va.us/>) offers job seekers several options for a self-directed job search in specific Virginia cities and counties, a national job search by state, Military Specialty, government jobs, and other national sites for jobs and employment information.

In its second year, CareerConnect (<http://www.careerconnect.state.va.us/default.htm>), the Commonwealth of Virginia's electronic one-stop workforce development system, is continuing to grow and reach an increasing number of citizens. The Virginia Employment Commission ([www.vec.state.va.us](http://www.vec.state.va.us)), and Local Workforce Investment Boards ([www.vec.state.va.us/pdf/wibcontact.pdf](http://www.vec.state.va.us/pdf/wibcontact.pdf)) have developed CareerConnect with a three-year grant Virginia from the U.S. Department of Labor.

On July 1, 2002, Virginia began a pilot that will allow job seekers to apply for all jobs online. The Mid-Atlantic Career Connect, or "MACC" will allow the job seeker to have his/her own personal identification number (PIN) and a folder dedicated and maintained on the system. This will be Virginia's most interactive job seeker site to date.

2. Virginia's Employment Commission's (VEC) online UI benefits filing system went live on May 6, 2002, and is available 24-hours a day, seven days a week. The service is a complete, end-to-end electronic process. In connection with developing the service, data collection was streamlined and redundancies in the B-10 paper form eliminated. This resulted in a more succinct, efficient online data collection instrument when compared to the paper B-10 form. No data is keyed or re-keyed by VEC personnel on claims filed via this online service.

The online UI benefits system was designed to permit claimants to save an incomplete filing for seven days before returning to complete and submit it. As a result, claimants who need to find their military pension number; employer's mailing address, or other such information need not lose all other data they have entered.

Since the service went live on May 6, through June 24, VEC customers filed 2,983 claims via the Internet, which represents about 6% of the total claims filed during that time period. Statistical reports indicate that the vast majority of Internet claims have originated from the Fairfax local field office in Northern Virginia, 1,136 claims, or 38% of the total number of Internet claims filed to date. Traffic conditions being what they are in Northern Virginia, VEC estimates the amount of time saved per claimant to be two hours in travel time to and from the Fairfax office even though round trip mileage may not be great. Similarly, many claimants in rural Virginia may drive an hour or more to reach a VEC local field office. VEC estimates the

total amount of saved claimant driving time for all online claims filed to date to be 5,966 hours.

In addition to driving time, the online UI benefits form is not just an exact duplicate of the B-10 paper form. In connection with developing the online service, data collection was streamlined and redundancies eliminated. Preliminary data suggests that it takes claimants an average of 8 minutes to complete and submit an online filing. This represents a substantial time saving over the time investment required to come into a field office, complete the claim form and meet with the interviewer.

3. The Department of Social Services, through its Child Support Enforcement Program, has used technology to make a significant difference to the delivery of human services in several areas.

The Child Support Enforcement Program has implemented an interactive Internet application <http://www.dss.state.va.us/family/dcse.html> to provide child support customers with information about their child support case. Customers can inquire about recent payments disbursed to them as well as the status of their case. The application, available since May 2001, now receives more than 3000 requests per day from child support customers.

The Child Support Enforcement Program uses technology to manage undistributed collections. Undistributed collections are a national problem in the child support program and result in child support payments being delayed. Virginia uses automated reports, procedures and on going monitoring and analysis to keep the percentage of undistributed collections below 3% of total collections. As a result of effective monitoring and control, Virginia processes and disburses 99% of all collections in 48 hours as required by federal law.

Virginia offers direct deposit to child support customers. Customers may choose to have their child support payments deposited into their savings or checking accounts. Direct deposit gets child support to customers faster and more securely than paper checks. More than 45% of disbursement transactions are now transmitted electronically.

4. Since the launch of the Virginia Department of Taxation (TAX) web site in 1997, TAX has completed a series of Internet applications designed to offer the convenience and speed of electronic services to the public. TAX now offers a full suite of online services for businesses and individual income tax customers. TAX and their corporate partner, American Management Systems (AMS), combine a variety of best practices drawn from industry successes with the latest technological tools. "Placing the customer first in everything we do" is the underlying theme in the projects TAX is undertaking now and in the future. A key objective is to create a dynamic environment for taxpayers to easily understand tax requirements and to file and pay in a timely, efficient, and convenient manner.

These Internet applications collectively referred to as VATAX Online, represent possibly the most complete suite of electronic services provided by any revenue agency. In the two years since the first pilot program, VATAX Online already accounts for more than 40% of all new



business registrations and has collected more than \$150 million in business tax payments. The individual income tax filing application is estimated to account for nearly 4% of all individual income tax returns filed in 2002 during just its first full year of availability. Because of the ability of these applications to provide fast and error-free electronic services, and their early adoption and acceptance by the public, VATAX Online is an excellent example of Virginia's focus on eGovernment services.

### **Virginia Business Community**

The Secretary of Technology's Web site notes that Virginia, in addition to being heralded as the Internet Capital of the World, is a major technology center for several industries, including semiconductor and other electronics manufacturing, biotechnology, selected segments of commercial aerospace, systems integration, advanced materials, intelligent transportation systems and wireless communications.

The Governor's Commission on Information Technology, composed primarily of IT business leaders, is looking at the issues of Internet policy; regional technology investment; workforce development; and tax and regulatory structures. Additionally, former Governor Gilmore chaired the congressional Advisory Commission on Electronic Commerce. Virginia resources, both public and private, provide assistance to the IT business community. They include:

- The [Innovative Avenue](#) web site is being built to bring Virginia technology users, businesses and events together in an innovative clearinghouse atmosphere.
- The [Office of Science and Technology](#) brings funding opportunities and training to new technology projects and to technology business startups.
- The [Center of Innovative Technology](#) increases the Commonwealth's economic competitiveness and quality of life by advancing the development of Virginia as a technology state and by creating and retaining technology-based jobs and businesses.
- [VirginiaLink](#) is a multi-vendor, multi-services telecommunications marketplace that enables businesses, which join the program, discounted rates and favorable terms for advanced telecommunications services.
- The [Virginia Electronic Commerce Technology Center](#) assists small businesses, local governments and regional agencies with e-commerce training and consulting, web site design and development, online ordering, and education programs.

1. In Virginia, businesses are required to register and file most of their taxes with Virginia Department of Taxation (TAX). VATAX Online for Businesses is a suite of Internet services that provides unprecedented flexibility and access to account information for business customers.

In an unprecedented intergovernmental project, Virginia became the first state to offer integrated online tax services for state employment taxes. TAX entered an agreement with the Virginia Employment Commission (VEC) to jointly develop, maintain and manage a suite of Internet services for the employer community.

As of July 2001, TAX's online business registration service (iReg) was integrated with VEC, enabling businesses to register for VEC taxes at the same time they register with TAX. This allows taxpayers to register for several taxes with two different state agencies simultaneously, eliminating a significant amount of redundant work and maintenance for the taxpayer. The business tax filing and payment application (iFile for Businesses) is also integrated with VEC to allow taxpayers to file and pay unemployment insurance taxes at the same time as they file and pay their taxes with TAX. This intra-government cooperation provides taxpayers with "one stop shopping" when it comes to paying business taxes.

TAX offers a comprehensive suite of customer-focused Internet tools to business taxpayers. A new business in Virginia can now register, file their returns, pay their taxes, and get assistance with account-related problems without filling out a single form or ever speaking to a Customer Service Representative.

2. VirginiaScan (<http://www.yesvirginia.org/vascan.asp>) is a web-enabled site selection application that was originally developed as a marketing tool and as an opportunity to streamline the location process for new or expanding businesses in Virginia. Since its launch in October 2000, the web site has provided a very effective approach to marketing Virginia because it provides a query tool on basic site selection information and at the same time reinforces the concept that Virginia is the "Digital Dominion".

The VirginiaScan application provides prospects a "first cut" opportunity to compile a list of those facilities in Virginia that meet their criteria. These criteria can be queried and results demonstrated in tabular and cartographic form on VirginiaScan. While the industrial site selection process is not something that can (or should) be completed entirely over the Internet, this tool allows Virginia to stay in consideration until a face-to-face relationship between the prospect and the Commonwealth can take place. This provides Virginia a competitive advantage and the opportunity to be "first to market" with accurate up-to-date information on sites, buildings, labor, utilities, transportation, and scores of other critical factors.

Although originally designed for marketing, the application has evolved into an efficient data maintenance tool as well. Virginia wants to be first to market with accurate answers to prospects' questions. With this in mind, the Virginia Economic Development Partnership (VEDP) decided to take an additional step and develop an on-line interactive data management tool. VirginiaScan Phase 3 (<http://virginiascan.yesvirginia.org/website/sites/redp/login.cfm>) is an interactive tool that allows Virginia's local and regional economic development allies a secured login to the VEDP computer system so that data describing their locality can be added and/or updated. This application went live on January 16, 2002.

Site Selection magazine, a widely read economic development trade magazine, recently announced that for the second year in a row the Virginia Economic Development Partnership (VEDP) has been named a "Top Development Group" for 2001. In addition, Virginia's 2001 Ford announcement of its \$375 million expansion in Norfolk was recognized as one of the "Top Deals of 2001."

VEDP won the honors based on new capital investment and new jobs, plus per-capita capital investment, and per-capita jobs. Site Selection also reviewed each nomination for evidence of new, value-adding services and programs to benefit prospective new firms and existing companies, as well as leadership, problem solving, innovation and cooperation. Site Selection cited such innovations as VDEP's newly redesigned web site and economic development legislation initiatives.

Additional information on Virginia technology achievements in the business community may be found <http://www.technology.state.va.us/TechVA/techinvabiz.cfm> on the Secretary of Technology's Web site.

## **Virginia Academia**

### K-12 Education:

Since 1995 Virginia has been committed to the Standards of Learning program emphasizing the need for improving public education in four core subjects - English, mathematics, science, and history and social science. Virginia is proud of the results of this program, especially the improvements in student achievement and the number of schools meeting state accreditation standards.

Virginia is one of three states recognized by the National Education Goals Panel for leading the nation in a key measure of student achievement-the increase in the number of students qualifying for college credit on Advanced Placement examinations.

#### The SOL Technology Initiative

(<http://www.pen.k12.va.us/VDOE/Technology/soltech/soltech.html>) will provide access to high stakes, standards of learning on-line testing in all high schools by 2004. Approximately 90 out of 132 school districts are ready to conduct on-line testing at high schools. The initiative will be extended to middle and elementary schools for completion in 2006 and 2009 respectively.

The goal of the program is to provide on-line access to instructional, remedial and assessment resources. Evidence of meeting the goal will include achieving a five-to-one student to computer ratio, establishment of high speed, high bandwidth networks in all schools, connectivity in all schools and creation and implementation of an assessment delivery system.

In the 2000 session of the General Assembly, legislation tied E-Rate funding to the Standards of Learning Technology Initiative to bring a higher level of connectivity to Virginia schools. E-Rate funding commitments have averaged \$25 million statewide in the first three years of the program, and discounts that average sixty percent on local and long distance telephone services and Internet access help to stretch already tight school budgets.

The achievements resulting from the Standards of Learning program are also reflected in other important ways. Results of the National Assessment of Educational Progress, the latest SAT tests, and the number of students taking Advanced Placement courses show substantial progress in students' achievement in public schools. Virginia ranks first in the South and tenth in the

nation in the percentage of students taking the SAT-1 that measures the verbal and mathematics abilities for successful performance in college.

The rewards public education provides for each student in Virginia public schools are vitally important to our society in a new and increasingly complex century. We must continue to strive to meet the educational needs of students in the commonwealth now and in the future.

#### Higher Education:

The following primary duties and responsibilities of the State Council of Education for Virginia (SCHEV) are to develop "policies, formulae, and guidelines for the fair and equitable distribution and use of public funds among [Virginia's] public institutions of higher education." In addition, they are also directed to analyze each institution's operating and capital budget request and provide recommendations to the Governor and the General Assembly regarding the approval or modification of each request.

The Commonwealth of Virginia has the 11th largest higher education system in the U.S. That system includes approximately:

- 372,000 students
- 314,000 in public institutions
- 50,000 in private not-for-profit institutions
- 8,000 in other institutions

Virginia has 15 public four-year institutions, 40 private not-for-profit institutions, 12 private for-profit, and 24 public two-year institutions, which includes: 23 community colleges on 39 campuses and one junior/transfer-oriented college. Virginia's public institutions award about 44,000 degrees annually. They can be broken down as follows:

- 11,000 Associate's degrees
- 31,000 Bachelor's degrees
- 11,000 Master's degrees
- 1,000 Doctoral degrees
- 1,900 Professional degrees

Over \$3 billion is spent on higher education in Virginia each year. About \$1.3 billion is provided through the state's General Fund and approximately \$869 million is provided through tuition and fees collected by the institutions. Over \$500 million is provided through federal and private sources.

Highlights of Virginia's education initiatives that are currently underway are listed below.

- Electronic Campus of Virginia (<http://www.e-cva.org/>) is a cooperative instructional technology initiative among the state's public and private colleges and universities. Twenty-three colleges and universities currently participate in this initiative, offering both undergraduate and graduate level courses. The Electronic Campus of Virginia web

site lists distributed or distance learning courses currently available statewide, complete with course schedules and delivery methods. A primary goal of the electronic campus is to provide students of all ages with "one-stop" access to learning in a distributed environment that meets their needs for convenient and high quality undergraduate, graduate and professional, and continuing education.

Electronic Campus of Virginia is one of several collaborative projects now underway among the state's colleges and universities to address workforce preparation challenges and to help meet needs for advanced education and technological literacy.

Electronic Campus of Virginia brings college courses from across Virginia as close as a personal computer. Students can:

- Identify courses or programs that are electronically delivered in multiple data formats. Among the data formats are: World Wide Web, satellite, compressed video and CD-Rom.
- Search by college or university, subject, or data format for more detailed information including course descriptions and how the courses are delivered.
- Connect directly to the college or university to learn about registration, enrollment, course description and cost.

The sponsoring accredited college or university is responsible for all procedures related to:

- Admissions
  - Credit Transfer
  - Delivery Format
  - Financial Aid
  - Prerequisites and any other course enrollment matters
  - Refund policies
  - Registration
  - Tuition and fees
- 
- TELETECHNET. Old Dominion University (ODU) offers the TELETECHNET program, <http://www.odu.edu/webroot/orgs/ao/dl/teletechnet.nsf>, which permits distance learning for students at one of the 50 sites within Virginia or five other states. TELETECHNET offers bachelor's and master's degree programs and is designed for part-time students. Cited by the Center for Digital Government in the 2001 State-by-State Results, TELETECHNET has exceeded growth expectations since its inception in 1992. ODU also participates in the Electronic Campus of Virginia program.
  - Virginia's community colleges (VCCS) offer a number of different ways to learn, and earn academic credit, without having to attend a traditional class on campus. By using different distance learning technologies (<http://www.vccs.cc.va.us/vccsonline/about.html>) such as the World Wide Web, e-mail,

videotapes, compressed video, telecourses, or audio conferencing, students can connect to courses and programs at Virginia's community colleges from home, from work, from their local library, from another college university, or even at the beach! VCCS is a partner in Electronic Campus of Virginia, TELETECHNET and the Electronic Campus of the Southern Regional Education Board described below.

- The *Electronic Campus* of the Southern Regional Education Board (SREB) (<http://www.electroniccampus.org/>) is an “electronic marketplace” for courses, programs and services. All courses and programs are offered by accredited colleges and universities in the SREB states and meet the *Principles of Good Practice* developed by the *Electronic Campus*. Virginia is a member of SREB and offers the electronic campus as a facet of its comprehensive distance learning initiatives.

In conclusion, the State Council of Higher Education for Virginia (SCHEV) (<http://www.schev.edu/>), in addition to the state portal, offers a list of colleges and universities under each of the primary headings on its web site:

- Students
- Parents
- K-12 Educators
- Administrators/Faculty
- Policymakers
- Reports/Statistics

The SCHEV web site acts as a portal with direct links to all public and private institutions of higher education licensed to do business in Virginia. The SCHEV web site also provides a tremendous range of higher education information designed for targeted audiences including students and parents. Among the more interesting services on the site is the official degree inventories for each public institution, complete with five years of fall enrollment history and degree award history by program and level (<http://research.schev.edu/degreeinventory/>). In the same folder, the SCHEV Program Finder (<http://research.schev.edu/degreeinventory/>) provides a side-by-side comparison of the degree programs available at each of the fifteen public four-year institutions in the state. A similar page for the Virginia Community College System is planned for late summer 2002.

The SCHEV web site also provides necessary information on financial aid, domicile policies and the Reports of Institutional Effectiveness – the only truly web-based set of accountability reports in the nation - which are truly best of breed themselves.

## Appendix D: Web Services Project Nomination Form & Evaluation Script

---

### Council On Technology Services 2002 Web Services Ad Hoc Workgroup Web Services Proof-of-Concept Project Nomination Form

**Nominee Name:**

**Job Title:**

**Organization:**

**Secretariat or Area:**

(e.g., Secretary of Education, Local Government, Independent Agency, etc.)

**Current Job**

Senior Management ☐ Technician ☐ Midlevel Management ☐ Other ☐  
Business Analyst ☐

**Primary Expertise** (check one)

Business ☐ Technology ☐

**Business Expertise** (check all that apply)

Strategic Planning ☐ Financial Analysis ☐  
Project Management ☐ Communications ☐  
Core Business Activities: \_\_\_\_\_

**Technology Expertise** (check all that apply)

Application Development ☐ Java and/or J2EE ☐  
Middleware Integration ☐ Web Enablement ☐  
Platform Configurations ☐ XML ☐  
Systems Management Support ☐ SOAP, WSDL, or UDDI standards ☐  
IT Security ☐ Distributed Development Concepts ☐

**Other Recommended Skills** (check all that apply)

Adapts well to new concepts ☐  
Works well in a team environment ☐



**COTS Member Information**

Recommended by (print name)

Title:

Comments:

Signature: (Not required if e-mailed)

Please send completed forms to [plubic@ntp.state.va.us](mailto:plubic@ntp.state.va.us) or by fax to the attention of Paul Lubic at (804) 371-2795, no later than February 15, 2002.

---

## Web Services Pilot Project Evaluation Script

### Approach

The project consists of several business partner solutions and applications from state agencies. Agency participants will develop the web services with support from the business partner participants. The development efforts will be documented in a standard format for comparison and completeness. A final report will be developed and submitted to COTS at the end of the project.

### Evaluation Script

The evaluation “script” to be used during the development process will be as follows:

1. Use of the development tool set
2. Developer skill set required
3. Product stability
4. Product’s adherence to standards
5. Speed of development
6. Quality of developed service, e.g., does it perform intended purpose
7. Completeness of product offering, e.g., uses UDDI, SOAP, WSDL and XML; has development tools
8. Security of the web service environment
9. Estimated cost to implement



## Appendix E: Glossary of Terms

(Sources: OASIS, W3C and What Is? <http://whatis.techtarget.com/>)

### ACORD

Association for Cooperative Operations Research and Development is a global, nonprofit insurance association.

### ADpr

Active Digital Profile specification, submitted to the OASIS Provisioning Services Technical Committee for advancement as part of PSML

### ANSI

American National Standards Institute

### API

Application Program Interface is the interface (calling conventions) by which an application program accesses operating system and other services. An API is defined at source code level and provides a level of abstraction between the application and the kernel (or other privileged utilities) to ensure the portability of the code.

### Array

A collection of identically typed data items distinguished by their indices (or "subscripts"). The number of dimensions an array can have depends on the language but is usually unlimited.

### ARTS

The Association for Retail Technology Standards is a division of the National Retail Federation.

### ASC X12

The Accredited Standards Committee (ASC) X12, accredited by the American National Standards Institute and comprised of cross-industry representation, develops robust e-business exchange specifications and electronic data interchange standards.

### AuthXML

Security specification submitted to the OASIS Security Services Technical Committee for advancement as part of SAML

### BTP

Business Transactions Protocol, an XML-based protocol for managing complex B2B transactions over the Internet

### CALS

Continuous Acquisition and Life-Cycle Support, a US Department of Defense initiative to improve weapon system acquisition and life-cycle support processes through accelerated creation and application of digital product data and technical information

CBEFF

Common Biometric Exchange File Format

CGM

Computer Graphics Metafile

CIQ

Customer Information Quality, a family of XML specifications for customer profile/information management

CRML

Customer Relationship Markup Language, an XML vocabulary specification that defines customer relationships, submitted by MSI Business Solutions to the OASIS CIQ Technical Committee

DocBook

XML/SGML vocabulary particularly well suited to books and papers about computer hardware and software

DSML

Directory Services Markup Language, an XML specification for marking up directory services information

DTD

Document Type Definition, the building blocks of an XML document

ebXML

Electronic Business Extensible Markup Language, sponsored by UN/CEFACT and OASIS, a modular suite of specifications that enable enterprises of any size and in any geographical location to conduct business over the Internet

ebXML CPA

ebXML Collaboration Protocol Agreements

ebXML CPP

ebXML Collaboration Protocol Profiles

ebXML CPPA

ebXML Collaboration Protocol Profile and Agreement specifications that define CPPs and CPAs

ebXML IIC

OASIS ebXML Implementation, Interoperability, and Conformance Technical Committee working to facilitate the creation of interoperable ebXML infrastructures and applications

### ebXML MSG

ebXML Messaging Services Specification, which provides a secure method for exchanging electronic business transactions using the Internet

### ebXML RegRep

ebXML Registry Repository Services Specifications, which define interoperable registries and repositories with an interface that enables submission, query and retrieval on the contents of the registry and repository

### ebXML RIM

ebXML Registry Information Model, which provides information on the types of metadata stored in an ebXML Registry as well as the relationships among the various metadata classes

### ebXML RS

ebXML Registry Services Specification, which defines the interface to ebXML Registry Services and identifies message definitions and XML schema

### ebXML TRP

ebXML Transport, Routing and Packaging Specification, see ebXML Messaging

### EIDX

The Electronics Industry Data Exchange Association is a section of the Computing Technology Industry Association (CompTIA) that represents the majority of companies in the electronics industry.

### EML

Election Markup Language, a specification for exchanging election and voter services information using XML

### Fragment Interchange

An OASIS (SGML Open) resolution that defines a method of exchanging portions of an SGML document

### GeoLang

Geography and Languages, a set of topic maps published subjects defining language, country, and region

### HTTP

Hypertext Transport Protocol is the set of rules for exchanging files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web

### HumanML

Human Markup Language, a specification designed to represent human characteristics through XML

### IEC

International Electrotechnical Commission; member of MoU on E-Business that also includes OASIS, ISO, ITU and UN/ECE

### IPR

Intellectual Property Rights

### ISO

International Organization for Standardization; member of MoU on E-Business that also includes OASIS, IEC, ITU, and UN/ECE

### ITML

Information Technology Markup Language, submitted to the OASIS Provisioning Services Technical Committee for advancement as part of PSML

### ITU

International Telecommunication Union; member of MoU on E-Business that also includes OASIS, IEC, ISO and UN/ECE

### Loosely Coupled

The method of N-tier application development where services and components are implemented as separate tiers. If a tier changes, it is easy to deploy the change since it does not affect other tiers.

### MoU on E-Business

Memorandum of Understanding between international standards-setting bodies to coordinate the development of global standards for electronic business; members include IEC, ISO, ITU, UN/ECE and OASIS

### OASIS

The Consortium is international; jointly hosted by the MIT Laboratory for Computer Science in the United States and in Europe by INRIA who provide both local support and performing core development. The W3C was initially established in collaboration with CERN, where the Web originated, and with support from DARPA and the European Commission."

### OSI

Open Systems Interconnection is a standard description or "reference model" for how messages should be transmitted between any two points in a telecommunication network. Its purpose is to guide product implementors so that their products will consistently work with other products. The reference model defines seven layers of functions that take place at each end of a communication. Although OSI is not always strictly adhered to in terms of keeping related functions together in a well-defined layer, many if not most products involved in telecommunication make an attempt to describe themselves in relation to the OSI model. It is also valuable as a single reference view of communication that furnishes everyone a common ground for education and discussion.

### RELAX

Regular Language description for XML, a language for validating XML documents that was merged with TREX to create RELAX NG

### RELAX NG

Regular Language Expression, a lightweight XML language validation specification based on TREX and RELAX

### S2ML

Security specification submitted to the OASIS Security Services Technical Committee for advancement as part of SAML

### SAML

Security Assertion Markup Language is an Extensible Markup Language (XML) standard that allows a user to log on once for affiliated but separate Web sites. SAML is designed for business-to-business (B2B) and business-to-consumer (B2C) transactions. SAML specifies three components: assertions, protocol, and binding. There are three assertions: authentication, attribute, and authorization. Authentication assertion validates the user's identity. Attribute assertion contains specific information about the user. And authorization assertion identifies what the user is authorized to do.

### SOAP

SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses. SOAP can potentially be used in combination with a variety of other protocols; however, the only bindings defined in this document describe how to use SOAP in combination with HTTP and HTTP Extension Framework.

### SPML

Service Provisioning Markup Language, an XML-based framework specification for exchanging user, resource, and service provisioning information

### TModel

A TModel is the UDDI definition of an Operation in the WSDL

### TREX

Tree Regular Expressions for XML, a language for validating XML documents that was merged with RELAX to create RELAX NG

### UBL

Universal Business Language, a standard library of XML business documents

### UDDI

Universal Description, Discovery and Integration is the directory technology used by service registries that allows the directory to be searched for a particular Web service. In essence, UDDI is a White Pages or Yellow Pages that can be used to locate Web Services. There can be both private and public UDDI directories.

#### UN/CEFACT

United Nations Centre for Trade Facilitation and Electronic Business, cosponsors with OASIS of ebXML

#### UN/ECE

United Nations Economic Commission for Europe; member of MoU on E-Business that also includes OASIS, IEC, ISO and ITU

#### URL

Universal Resource Locator is a standard way of specifying the location of an object, typically a web page, on the Internet. Other types of object are described below. URLs are the form of address used on the World-Wide Web. They are used in HTML documents to specify the target of a hyperlink, which is often another HTML document (possibly stored on another computer).

#### WebCGM

An application of the ISO-standard Computer Graphics Metafile for electronic documents

#### Web Services

The Web Services Architectural Requirements Working Group of the World Wide Web Consortium (W3C), which develops standards for interoperable technologies, defines Web Services to be a software application identified by a URI, whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via internet-based protocols.

#### W3C

"The World Wide Web Consortium exists to realize the full potential of the Web.

The W3C is an industry consortium which seeks to promote standards for the evolution of the Web and interoperability between WWW products by producing specifications and reference software. Although W3C is funded by industrial members, it is business partner-neutral, and its products are freely available to all.

#### WSCM

Web Services Component Model for Interactive Applications, see WSIA

#### WSDL

Web Services Description Language is the language used to create service descriptions of Web Services. It can be used to describe the location of the service and how to run it, as well as what business is hosting the service, the kind of service it is, keywords associated with the service and similar information.

### WSIA

Web Services for Interactive Applications, a coordinated set of XML vocabularies and Web services interfaces that allow companies to deliver Web applications to end users through a variety of channels

### WSRP

Web Services Standard for Remote Portals, an XML and Web services specification that will allow for the "plug-n-play" of visual, user-facing Web services with portals or other intermediary Web applications

### WSUI

Web Services User Interface, a specification for defining standard user interfaces for Web services, submitted to the OASIS WSIA Technical Committee

### WSXL

Web Services Experience Language, a Web services-centric component model, submitted by IBM to the OASIS WSIA Technical Committee for advancement as WSIA

### XACML

eXtensible Access Control Markup Language, an XML specification for expressing policies for information access over the Internet

### xAL

Extensible Address Language specification

### XBRL

XBRL is an international consortium comprising accounting standards bodies, accounting firms, technology companies, financial information providers, corporations, and government regulatory bodies.

### XCBF

XML Common Biometric Format, an XML schema for describing information that verifies identity based on human characteristics such as DNA, fingerprints, iris scans, and hand geometry.

### xCBL

XML component and business document library submitted to the OASIS UBL Technical Committee by Commerce One

### xCIL

Extensible Customer Information Language specification

### xCRL

Extensible Customer Relationships Language

XLIFF

XML Localization Interchange File Format, an XML specification for multi-lingual data exchange

xNAL

Extensible Name and Address Language specification

xNL

Extensible Name Language specification

XML

Extensible Markup Language is a markup language that structures information so that the information can be easily extracted and used by other applications. It uses tags, as does HTML, but those tags are used to structure and define information rather than display it. Service descriptors that detail how Web Services can be located and run are written in XML.

XMLvoc

Vocabulary for XML Standards and Technologies, a set of topic maps published subjects defining the vocabulary of interchangeable ontologies for the XML domain

XPath test

XML Path Language test, an OASIS conformance suite for XSLT/XPath processors

XRPM

Extensible Resource Provisioning Management specification, submitted to the OASIS Provisioning Services Technical Committee for advancement as part of PSML

XSLT test

XML Stylesheet Language Transformation test, an OASIS conformance suite for XSLT/XPath processors



## **Appendix F: Team 1 Final Report – DMV, Microsoft, & Susquehanna Technologies**

### **Web Services Proof-of-Concept Project Report**

#### **1. PROJECT PLAN (SCHEDULE)**

See attached project plan

#### **2. LIST OF TEAM NAME, MEMBERS, ROLES AND RESPONSIBILITIES**

DMSACWSWG

DMV, Microsoft, Susquehanna Technologies Address Change Web Service Work Group

##### **Team Members:**

<b>Name</b>	<b>Organization</b>	<b>E-Mail</b>	<b>Phone</b>
Chuck O’Keeffe	Microsoft Corporation	<a href="mailto:chuckok@microsoft.com">chuckok@microsoft.com</a>	(540) 247-8063
Hank Farlow	Microsoft Corporation	<a href="mailto:hankf@microsoft.com">hankf@microsoft.com</a>	(804) 307-9591
Beth DeHaven	Microsoft Corporation	<a href="mailto:bdehaven@microsoft.com">bdehaven@microsoft.com</a>	(804) 337-3262
William Strydom	DMV	<a href="mailto:dmvw1s@dmv.state.va.us">dmvw1s@dmv.state.va.us</a>	(804) 367-8402
Lana Shelley	DMV	<a href="mailto:dmvlss@dmv.state.va.us">dmvlss@dmv.state.va.us</a>	(804) 367-2635
Don Kendrick	DMV	<a href="mailto:dmvdsk@dmv.state.va.us">dmvdsk@dmv.state.va.us</a>	(804) 367-8336
Scott Fowler	VIPNET	<a href="mailto:sfowler@vipnet.org">sfowler@vipnet.org</a>	(804) 786-6220
Tim Mauney	DMV	<a href="mailto:Dmvt1m@dmv.state.va.us">Dmvt1m@dmv.state.va.us</a>	(804) 367-1379
David Froggatt	DMV	<a href="mailto:Dmvdcf@dmv.state.va.us">Dmvdcf@dmv.state.va.us</a>	(804) 367-2380
Bill Vencil	Susquehanna	<a href="mailto:Billv@susqtech.com">Billv@susqtech.com</a>	(540) 665 3427
Glenn Hickman	Susquehanna	<a href="mailto:Glennh@susqtech.com">Glennh@susqtech.com</a>	(540) 723 8700

##### **New members as of 7/1/2002:**

<b>William Xiaojin Xi</b>	<b>Microsoft Corporation</b>	<a href="mailto:williamx@microsoft.com">williamx@microsoft.com</a>	<b>(602) 315 0937</b>
---------------------------	------------------------------	--	-----------------------

##### **Left DMV effective 6/14/2002:**

Debbie Dodson	DMV	<a href="mailto:dmvdhd@dmv.state.va.us">dmvdhd@dmv.state.va.us</a>	(804) 367-9227
---------------	-----	--	----------------

##### **Roles and responsibilities**

Product Manager

- Holder of the project requirements

- Coordinates activities between development team and work-group
- Drives project vision
- Manages expectations
- Drives feature identification and prioritization
- Develops, maintains, and executes the communications plan

#### Program Manager

- Manages overall process
- Manages resource allocation
- Manages project schedule and status reporting
- Manages project scope and specification
- Facilitates team communication and negotiation
- Drives overall critical trade-off decisions

#### Developer

- Builds and tests features to meet the specifications and expectations
- Participates in design
- Estimates time and effort to complete each feature
- Servers the team as a technology consultant

#### Tester

- Develops testing strategy, plans, and scripts
- Manages the build process
- Conducts tests to accurately determine the status of product development
- Participates in setting the quality bar

#### Logistics Management

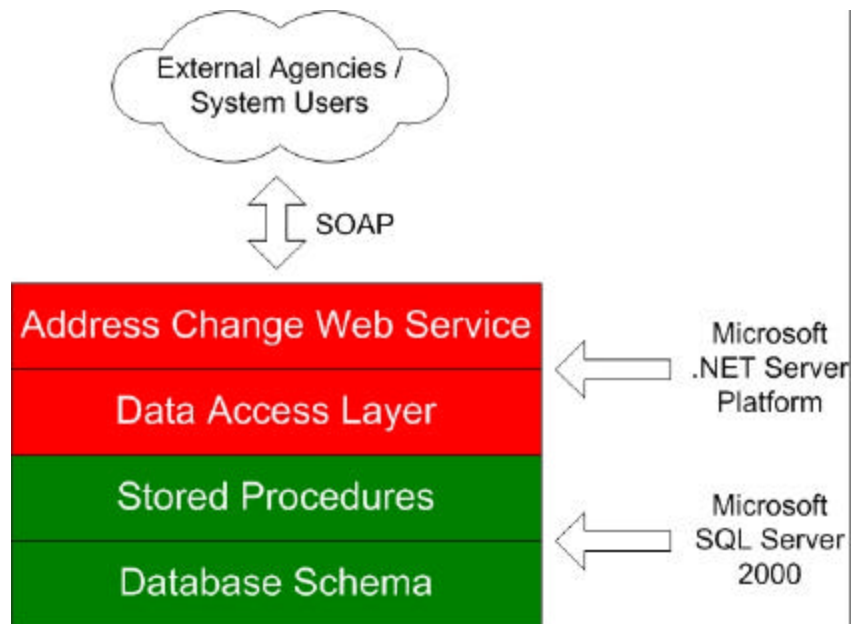
- Acts as team advocate to operations
- Acts as operations advocate to the team
- Plans and manages product deployment
- Participates in design, focusing on manageability, supportability, and deployability
- Supports the product during testing

#### Analyst

- Documents technical architecture of the development process
- Identifies and documents technical requirements outside of the scope of current development process
- Completes final report

### 3. PROOF-OF-CONCEPT DESIGN

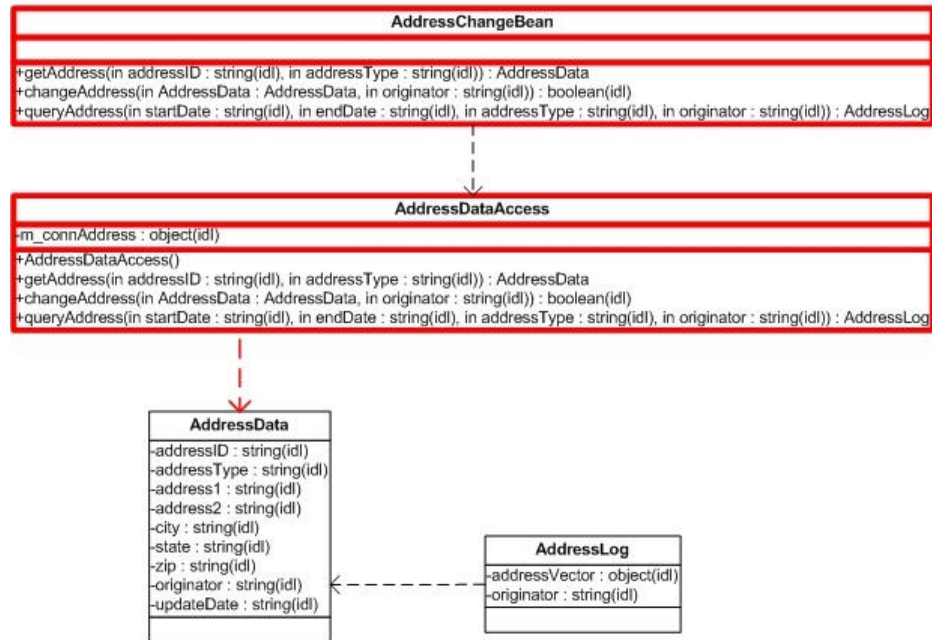
**Web Service Component:** The overall design of the team's Web Services application was dictated by a common WSDL (Web Service Definition Language) file agreed upon by the development teams of the workgroup. The system architecture is straightforward, utilizing Microsoft's .NET Framework to house and operate the web services component. Here is an overview diagram of the technical architecture:



Following the specifications of the common WSDL, a Web Service component was created. It is comprised of 3 members that perform the following:

- getAddress() – retrieves address information for a given person
- changeAddress() – changes address for a given person or persons
- queryAddress() – retrieves all address changes within a given time period

A simple UML model on the next page provides a more detailed view of the component:



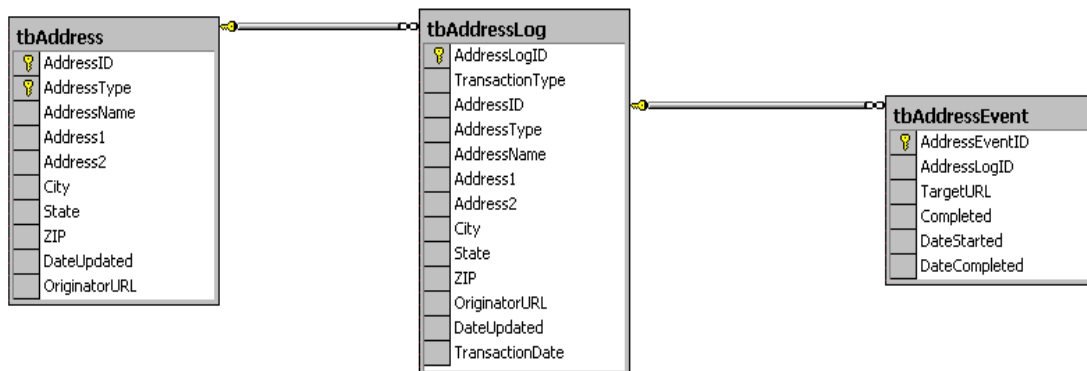
In addition to the Web Service component itself, the following modules were designed and implemented in order to complete the project:

**Database:** A database schema was designed and implemented as a database in the Microsoft SQL Server 2000 environment. Its purpose is to store all address information and log all address additions, deletions and changes. There are three tables in the schema along with associated stored procedures:

tbAddress: Repository for base address information

tbAddressLog: Maintains address change transaction for verification

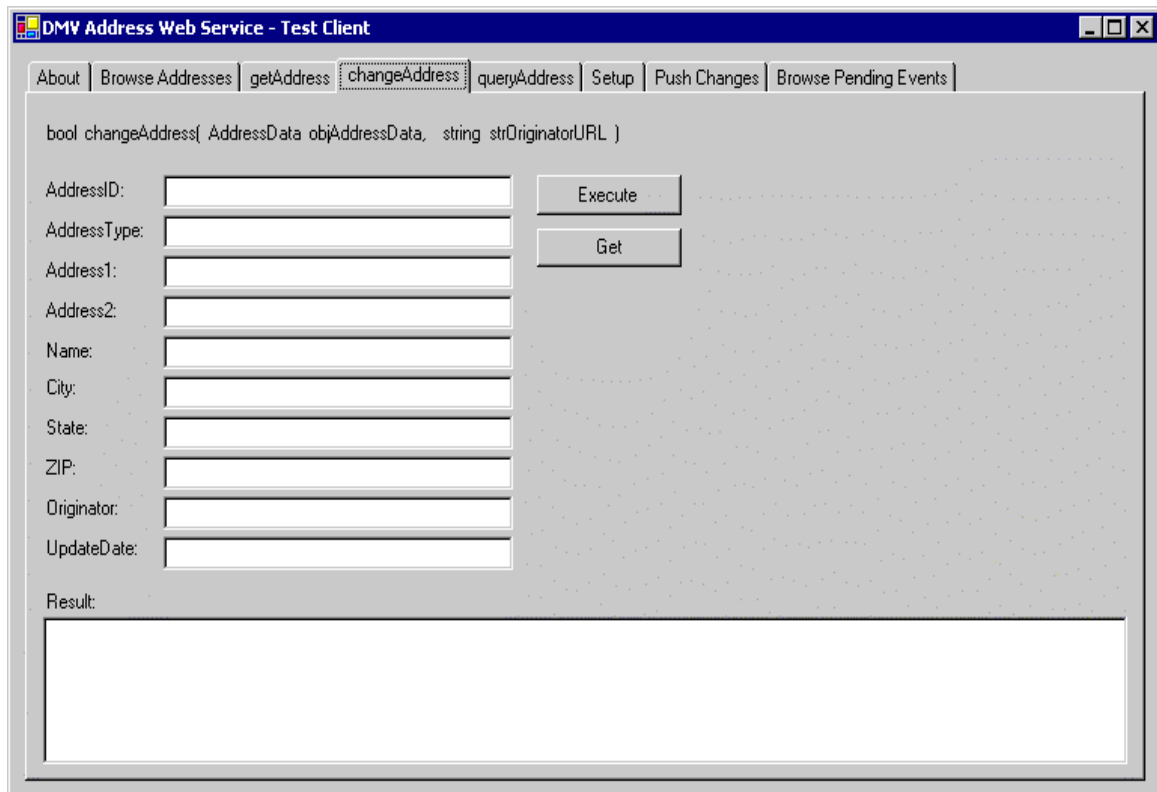
tcAddressEvent: Address change events that must be processed by background process.



**Background process:** In order to automate the process of updating and querying the other teams' exposed web services, a scheduled background process has been designed and implemented. Its purpose is to deliver all queued address changes events to each

address change Web Service found on the project UDDI server. The process is configured to execute every 15 minutes, pushing all queued address changes to the other systems.

**Client Test Application:** In order to more easily test the functionality of the web service and manually create address changes, a desktop application was developed using Visual Studio .NET:



#### 4. BUSINESS/FUNCTIONAL REQUIREMENTS

- Provide an address change function, which will utilize web services in the background to update subscribers of the service of the address change information.
- Address change information will be provided (pushed) to all subscribers in a **change address** function, if the change originated at the DMV.
- A **get address** function will be provided, if a subscriber chooses to retrieve (pull) the address information.
- A **query address changes** function will also be provided. A web service subscriber may choose to retrieve (pull) all address changes performed between specified date/time ranges for a specific agency/locality (originator).
- Only address changes that originated at the DMV will be propagated to all subscribers. This will prevent recursive looping of address change processing with other agencies/localities.

## 5. PROJECT EQUIPMENT DESCRIPTION

### Hardware - provided by Compaq

- 2 Compaq Proliant Servers ML Series
- (2) 72 GB Hard Drives
- 512 MB Memory
- Internal tape
- 2 – 17” monitors
- 1 Workstation for application development and testing of remote access to web services

### Software

- Visual Studio.NET – Web Services Development
    - All components for XML, WSDL, SOAP, UDDI
  - Windows 2000 Advanced Server for hosting platform. Also included in the installation is the .NET Framework to provide Web Services capabilities
  - SQL Server 2000 for database storage and analytic processing of data
- Microsoft provided the software and hardware.

### Location

- The servers are housed at VIPNET.

## 6. ACQUISITION AND INSTALLATION ACTIVITIES

Equipment was acquired by Microsoft for the purposes of the pilot project.

## 7. SERVER REQUIREMENTS

There are no additional requirements above and beyond the equipment described above.

## 8. NETWORKING REQUIREMENTS

There are no specific networking requirements other than the following:

- Both Compaq servers must be on the same network segment.
- The web server must be available on the Internet allowing connections via the HTTP protocol and ports 80 and 443 (inbound and outbound).

## 9. TRAINING REQUIREMENTS

- Tim Mauney and William Strydom from the DMV were sent to the .NET training provided by Microsoft at DIT on June 6 & 7.
- Microsoft gave DMV over fifteen reference books on various Microsoft products on August 2, 2002.
- No other further is required for this project.

## 10. TESTING PLAN

- Please see attached testing plan as submitted to the workgroup.
- Also, please see attached sample of testing results.
- If necessary, we can provide all testing results from our log files in whatever format is deemed appropriate by the larger group.

## 11. DEVELOPMENT/TECHNICAL

### July 18<sup>th</sup> update

1. William Xi, a member of Microsoft's .NET Velocity team has reviewed all components of the application to ensure that the architecture follows Microsoft's recommended practices.
2. The development process is largely complete. The major issue remaining is to get completed UDDI specs from the UDDI team so that the background process can connect to the UDDI server to publish and query.
3. Tim Mauney and Dave Froggatt are tentatively planning to deploy all application components to the DMV test servers on or about 7/24.

### August 19<sup>th</sup> update

1. Development of all components is complete.
2. Occasional "code adjustment" has been ongoing through the testing process.
3. Code base fully deployed on test servers.
4. Servers up and running with web service components and background processes.

## 12. PROTOCOL/SPECIFICATION ISSUES

### June 20<sup>th</sup> update

When using Microsoft's WSDL utility, some minor errors were encountered when converting the workgroup WSDL that was provided by BEA. Some minor editing of the utility's output corrected the issue.

## 13. INTEROPERABILITY ISSUES

### June 20<sup>th</sup> update

The .NET-based Web Service has been successfully tested from a PERL environment using Soap Lite. Here is the sample code used and resulting output:

#### Callfromperl.pl Source Code:

```
use SOAP::Lite;
my $param1 = SOAP::Data->type(string => "2112")->name('string0');
my $param2 = SOAP::Data->type(string => "individual")->name('string1');
my $objResult = SOAP::Lite
-> uri('http://localhost/DMVAddressWebService')
-> on_action(sub{sprintf '%s/%s', @_ })
-> proxy('http://localhost/DMVAddressWebService/AddressChangeBean.asmx')
-> getAddress( $param1, $param2 )
```

```
-> result or die "Failed to attach to web service";
print "AddressID: $objResult->{addressID}\n";
print "AddressType: $objResult->{addressType}\n";
print "Address1: $objResult->{address1}\n";
print "Address2: $objResult->{address2}\n";
print "City: $objResult->{city}\n";
print "State: $objResult->{state}\n";
print "ZIP: $objResult->{zip}\n";
print "Originator: $objResult->{originator}\n";
print "Update Date: $objResult->{updateDate}\n";
```

**Output:**

```
C:\DATA\Projects\MCS\DMV>perl callfromperl.pl
AddressID: 2112
AddressType: individual
Address1: 3668 Apple Pie Ridge Road
Address2:
City: Winchester
State: VA
ZIP: 22601
Originator: http://localhost/DMVAddressWebService
Update Date: 2002-05-08T00:00:00.0000000-04:00
```

**July 18<sup>th</sup> update**

Further internal interoperability testing has been performed by the team and no outstanding issues have been identified. As of this date no workgroup level interoperability testing has been performed.

**August 19<sup>th</sup> update**

In general, the concept of interoperability has been moderately successful. The DMV team has successfully connected to the VABC, VRS and GMU systems. There have been errors returned upon occasion from these “sister” systems that our system is unable to interpret. We do not believe that this is a base technology issue, but rather a symptom of the pilot nature of the project. In a realistic production system, the error handling system would be much more robust.

The use of UDDI to provide dynamic subscriptions to other teams’ web services was unsuccessful. It has been generally acknowledged that the concept is feasible, but would require a level of effort that is outside the scope of this project.

## 14. “BEST PRACTICE” COMMENTS

- 6/20/02 - When converting a WSDL created from another development platform, manually verify that the WSDL generation was correct.
- 6/20/02 - Keep the solution as simple as possible while still meeting the design requirements. This will help speed up the process of development and guarantee success.
- 06/20/02 - Be aware that a Web Service is a loosely coupled, disconnected process. Session state is an issue that needs to be considered carefully. While a concern, this is very similar in concept to the general practice of web application development.



- 7/18/02 - General Microsoft platform “best practices” should be followed in any type of Microsoft-based development efforts. Microsoft has provided the .NET Architecture Center, a resource of concepts, patterns and best practices for developing in the .NET environment. It is located at <http://msdn.microsoft.com/architecture>.
- 7/18/02 - Naming of parameters in the WSDL should not use common terms that may be reserved words. For example “string” is commonly used as a data type.
- 8/19/02 - Robust exception handling should include the judicious use of SOAP exceptions. This would allow the calling process to receive more detailed information in the event of a system error.
- 8/19/02 – When creating a set of Web Service components, much of the upfront design time should be spent in determining the optimal programming interface to the components. Once implemented, the interface is very difficult to change without affecting subscribers to the Web Service.
- 8/19/02 - When passing XML-based data to a Web Service, the originating process should validate the data by comparing it to the appropriate XML schema or DTD.
- 8/19/02 – Change management processes should be utilized in order to avoid confusion between web service interfaces and function/data requirements.
- 8/19/02 – A common set of data for interoperability testing and web service cross communication is critical.
- 8/19/02 - Common data validation rules should be implemented, if at all possible, so data errors can be minimized/avoided when calling the provided services.

## 15. OTHER CONCERNS / ISSUES

- 6/20/02 - We would like to suggest that architects/developers from each team be encouraged to communicate details of implementation and deployment before interoperability testing begins. In addition, all teams should agree upon a common method of communication that keeps everyone “in the loop”.
- 6/20/02 - Specific UDDI configuration information is necessary to complete the background process application.
- 8/19/02 - The project has been somewhat complicated by the lack of strict adherence to the agreed upon WSDL. Different teams interpreted the use of the WSDL in several different ways. This lack of commonality increased the complexity of attaching to each service. It is important to note however, that this is not a limitation of the web services specification, but rather variations on the implementation of the WSDL.
- 8/19/02 - In general, the use of UDDI for this project has been more limited than initially conceived.

## 16. OTHER COMMENTS

- 8/19/02 - Web service standards are a viable technology can provide real world solution for businesses. The standards involved (XML, WSDL, SOAP, UDDI) do indeed

create an environment that enables effective cross platform/application communications. Visual Studio.NET proved to be a highly effective tool for building and deploying web services. We appreciated the exception/error handling capabilities of VS.NET when trouble-shooting.

- 8/19/02 - Even though UDDI was not as fully utilized as originally conceived, the Microsoft UDDI SDK was effective in publishing and searching for web services.
- 8/19/02 - Most of the issues that we encountered were not limitations of the Web Services specs or Microsoft .NET development tools, but rather had to do with software design and coordination that is normally found in any software development project.
- 8/19/02 - The developers involved (Tim Mauney, Dave Froggatt, Glenn Hickman, William Xi) were all able to effectively leverage their software development expertise in the web services arena.
- 8/19/02 - We very much appreciated the excellent level of coordination between the workgroup teams during the interoperability testing phase.
- 8/19/02 - The test harness was very, very helpful. Thanks Will!

## 17. COST/TIME ESTIMATE

### 1) Meetings:

05/15/2002 DMV & Microsoft workgroup	1½ hours at DMV
05/22/2002 DMV & Microsoft workgroup	1½ hours at DMV
05/29/2002 DMV & Microsoft workgroup	1½ hours at DMV
05/30/2002 All workgroups	8 hours at VRS
06/04/2002 DMV, Microsoft & Susquehanna workgroup	1 hour at DMV
06/10/2002 DMV & Susquehanna workgroup	2 hours at DMV
06/20/2002 All workgroups	4 hours at VRS
06/25/2002 DMV & Microsoft workgroup	2 hours at DMV
07/18/2002 All workgroups	4 hours at VRS
08/16/2002 DMV & Susquehanna workgroup	2 hours at DMV
08/19/2002 All workgroups	4 hours at DIT

### 2) Development:

06/20/2002 Susquehanna development completed	35 hours at SusQtech
07/15/2002 Susquehanna development and meetings	85 hours at ST and DMV
08/19/2002 Susquehanna development	54 hours at SusQtech and Microsoft

### 3) Training:

06/06/2002 .NET training	8 hours at DIT
06/07/2002 .NET training	8 hours at DIT

### 4) Interoperability Testing:

08/19/2002 Susquehanna interop testing	32 hours at SusQtech and Microsoft
--	------------------------------------

---

## DMV/Microsoft/Susquehanna Technologies test plan

### Test strategy:

- Component testing
- Integration testing scenarios

### Component testing:

1. Test Web Service components - All methods
2. Test Database components - All stored procedures
3. Test ability to connect to UDDI
4. Test ability to publish changes [changeAddress()]
5. Test queryAddress() function
6. Test getAddress() function
7. Test all functions using Client application
8. Verify logs are populated with changes
9. Ensure that no recursion occurs

### Integration testing Scenarios:

1. Developers on the team will perform unit testing. Minimal “after the fact” documentation. Process described above.
2. Customer changes address at DMV
  - a. Address at DMV was actually changed
  - b. The address is propagated to the subscriber list from UDDI.
  - c. Each subscriber confirms the receipt of the change.
  - d. Log results from address changes push (true/false/SOAP error)
3. DMV receives and address change from remote system
  - a. Apply change. Ensure that the change was successful
  - b. Ensure that the change is not propagated to the subscriber list.
4. Run getAddress() for core set of data
  - a. Must receive address information from all subscriber systems
5. Run getAddress() for data unique to the team
  - a. Get no information back if the address has never been propagated
  - b. Get information if address was previously propagated and was stored in remote system.
6. Run queryAddress() for specific sets of data on remote systems
  - a. Propagate 10 address changes to all subscribers with DMV as the originator.
  - b. Query all subscribers for date range of above changes with DMV as originator.
  - c. Should get back all 10 changes from all subscribers with the original address change information.
7. queryAddress() from another system
  - a. receive request
  - b. Retrieve address logs between dates for specified originator.

## Sample of DMV Web Services testing results from 8/8/02 to 8/9/02

```
----- Push Log Entry: 8/8/2002 4:58:31 PM
AddressID: [000005389 ] Target: [GMU]
----- Push Log Entry: 8/8/2002 4:58:32 PM
AddressID: [000005389]
----- Push Log Entry: 8/8/2002 4:58:32 PM
AddressType: [Business]
----- Push Log Entry: 8/8/2002 4:58:32 PM
AddressName: [Mr. Joseph H. Smith III]
----- Push Log Entry: 8/8/2002 4:58:32 PM
Address1: [280 Laurel Park Av]
----- Push Log Entry: 8/8/2002 4:58:32 PM
Address2: []
----- Push Log Entry: 8/8/2002 4:58:32 PM
City: [Montpelier]
----- Push Log Entry: 8/8/2002 4:58:33 PM
State: [VA]
----- Push Log Entry: 8/8/2002 4:58:33 PM
ZIP: [24112]
----- Push Log Entry: 8/8/2002 4:58:33 PM
UpdateDate: [7/26/2002 12:00:00 AM]
----- Push Log Entry: 8/8/2002 4:58:33 PM
Originator: [DMV]
----- Push Log Entry: 8/8/2002 4:58:33 PM
Calling changeAddress() on GMU.
----- Push Log Entry: 8/8/2002 4:58:34 PM
Received [True] from GMU.
----- Push Log Entry: 8/8/2002 4:58:34 PM
SUCCESS! (Pushed AddressLogID [11115] to GMU!
----- Push Log Entry: 8/8/2002 4:58:34 PM
Marked AddressEventID [10052] as completed.
----- Push Log Entry: 8/8/2002 4:58:34 PM
AddressID: [000005389 ] Target: [VRS]
----- Push Log Entry: 8/8/2002 4:58:35 PM
AddressID: [000005389]
----- Push Log Entry: 8/8/2002 4:58:35 PM
AddressType: [Business]
----- Push Log Entry: 8/8/2002 4:58:35 PM
AddressName: [Mr. Joseph H. Smith III]
----- Push Log Entry: 8/8/2002 4:58:35 PM
Address1: [280 Laurel Park Av]
----- Push Log Entry: 8/8/2002 4:58:35 PM
Address2: []
----- Push Log Entry: 8/8/2002 4:58:35 PM
City: [Montpelier]
----- Push Log Entry: 8/8/2002 4:58:35 PM
State: [VA]
----- Push Log Entry: 8/8/2002 4:58:35 PM
ZIP: [24112]
----- Push Log Entry: 8/8/2002 4:58:35 PM
UpdateDate: [7/26/2002 12:00:00 AM]
----- Push Log Entry: 8/8/2002 4:58:35 PM
Originator: [DMV]
----- Push Log Entry: 8/8/2002 4:58:35 PM
Calling changeAddress() on VRS.
----- Push Log Entry: 8/8/2002 4:58:46 PM
Received [True] from VRS.
----- Push Log Entry: 8/8/2002 4:58:46 PM
SUCCESS! (Pushed AddressLogID [11115] to VRS!
----- Push Log Entry: 8/8/2002 4:58:46 PM
Marked AddressEventID [10053] as completed.
----- Push Log Entry: 8/8/2002 4:58:46 PM
AddressID: [000005389 ] Target: [VABC]
```

```

----- Push Log Entry: 8/8/2002 4:58:47 PM
AddressID: [000005389]
----- Push Log Entry: 8/8/2002 4:58:47 PM
AddressType: [Business]
----- Push Log Entry: 8/8/2002 4:58:47 PM
AddressName: [Mr. Joseph H. Smith III]
----- Push Log Entry: 8/8/2002 4:58:47 PM
Address1: [280 Laurel Park Av]
----- Push Log Entry: 8/8/2002 4:58:47 PM
Address2: []
----- Push Log Entry: 8/8/2002 4:58:47 PM
City: [Montpelier]
----- Push Log Entry: 8/8/2002 4:58:47 PM
State: [VA]
----- Push Log Entry: 8/8/2002 4:58:47 PM
ZIP: [24112]
----- Push Log Entry: 8/8/2002 4:58:47 PM
UpdateDate: [7/26/2002 12:00:00 AM]
----- Push Log Entry: 8/8/2002 4:58:47 PM
Originator: [DMV]
----- Push Log Entry: 8/8/2002 4:58:47 PM
Calling changeAddress() on VABC.
----- Push Log Entry: 8/8/2002 4:58:48 PM
Received [True] from VABC.
----- Push Log Entry: 8/8/2002 4:58:48 PM
SUCCESS! (Pushed AddressLogID [11115] to VABC!
----- Push Log Entry: 8/8/2002 4:58:48 PM
Marked AddressEventID [10054] as completed.
----- Push Log Entry: 8/8/2002 4:58:48 PM
AddressID: [000005389 ] Target: [DHRM]
----- Push Log Entry: 8/8/2002 4:58:49 PM
AddressID: [000005389]
----- Push Log Entry: 8/8/2002 4:58:49 PM
AddressType: [Business]
----- Push Log Entry: 8/8/2002 4:58:49 PM
AddressName: [Mr. Joseph H. Smith III]
----- Push Log Entry: 8/8/2002 4:58:49 PM
Address1: [280 Laurel Park Av]
----- Push Log Entry: 8/8/2002 4:58:49 PM
Address2: []
----- Push Log Entry: 8/8/2002 4:58:49 PM
City: [Montpelier]
----- Push Log Entry: 8/8/2002 4:58:49 PM
State: [VA]
----- Push Log Entry: 8/8/2002 4:58:49 PM
ZIP: [24112]
----- Push Log Entry: 8/8/2002 4:58:49 PM
UpdateDate: [7/26/2002 12:00:00 AM]
----- Push Log Entry: 8/8/2002 4:58:49 PM
Originator: [DMV]
----- Push Log Entry: 8/8/2002 4:58:49 PM
Calling changeAddress() on DHRM.
----- Push Log Entry: 8/8/2002 4:58:54 PM
Received [False] from DHRM.
----- Push Log Entry: 8/8/2002 4:58:54 PM
FAILURE! (Failed to push AddressLogID [11115] to DHRM!
----- Push Log Entry: 8/9/2002 9:45:01 AM
AddressID: [000005389 ] Target: [DHRM]
----- Push Log Entry: 8/9/2002 9:45:03 AM
AddressID: [000005389]
----- Push Log Entry: 8/9/2002 9:45:03 AM
AddressType: [Business]
----- Push Log Entry: 8/9/2002 9:45:03 AM
AddressName: [Mr. Joseph H. Smith III]
----- Push Log Entry: 8/9/2002 9:45:03 AM
Address1: [280 Laurel Park Av]
----- Push Log Entry: 8/9/2002 9:45:03 AM
Address2: []
----- Push Log Entry: 8/9/2002 9:45:03 AM
City: [Montpelier]
----- Push Log Entry: 8/9/2002 9:45:03 AM

```

```

State: [VA]
----- Push Log Entry: 8/9/2002 9:45:03 AM
ZIP: [24112]
----- Push Log Entry: 8/9/2002 9:45:03 AM
UpdateDate: [7/26/2002 12:00:00 AM]
----- Push Log Entry: 8/9/2002 9:45:03 AM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 9:45:03 AM
Calling changeAddress() on DHRM.
----- Push Log Entry: 8/9/2002 9:45:08 AM
Received [False] from DHRM.
----- Push Log Entry: 8/9/2002 9:45:08 AM
FAILURE! (Failed to push AddressLogID [11115] to DHRM!
----- Push Log Entry: 8/9/2002 10:00:01 AM
AddressID: [000005389 ] Target: [DHRM]
----- Push Log Entry: 8/9/2002 10:00:03 AM
AddressID: [000005389]
----- Push Log Entry: 8/9/2002 10:00:03 AM
AddressType: [Business]
----- Push Log Entry: 8/9/2002 10:00:03 AM
AddressName: [Mr. Joseph H. Smith III]
----- Push Log Entry: 8/9/2002 10:00:03 AM
Address1: [280 Laurel Park Av]
----- Push Log Entry: 8/9/2002 10:00:03 AM
Address2: [ ]
----- Push Log Entry: 8/9/2002 10:00:03 AM
City: [Montpelier]
----- Push Log Entry: 8/9/2002 10:00:03 AM
State: [VA]
----- Push Log Entry: 8/9/2002 10:00:03 AM
ZIP: [24112]
----- Push Log Entry: 8/9/2002 10:00:03 AM
UpdateDate: [7/26/2002 12:00:00 AM]
----- Push Log Entry: 8/9/2002 10:00:03 AM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 10:00:03 AM
Calling changeAddress() on DHRM.
----- Push Log Entry: 8/9/2002 10:00:08 AM
Received [False] from DHRM.
----- Push Log Entry: 8/9/2002 10:00:08 AM
FAILURE! (Failed to push AddressLogID [11115] to DHRM!
----- DMVAddressPush : 8/9/2002 10:30:02 AM
AddressID: [000005361 ] Target: [GMU]
----- Push Log Entry: 8/9/2002 10:30:03 AM
AddressID: [000005361]
----- Push Log Entry: 8/9/2002 10:30:03 AM
AddressType: [Business]
----- Push Log Entry: 8/9/2002 10:30:03 AM
AddressName: [Mr. James Smith]
----- Push Log Entry: 8/9/2002 10:30:03 AM
Address1: [3668 Apple Pie Ridge Road]
----- Push Log Entry: 8/9/2002 10:30:03 AM
Address2: [ ]
----- Push Log Entry: 8/9/2002 10:30:03 AM
City: [Winchester]
----- Push Log Entry: 8/9/2002 10:30:03 AM
State: [VA]
----- Push Log Entry: 8/9/2002 10:30:03 AM
ZIP: [22603]
----- Push Log Entry: 8/9/2002 10:30:03 AM
UpdateDate: [8/8/2002 12:00:00 AM]
----- Push Log Entry: 8/9/2002 10:30:03 AM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 10:30:03 AM
Calling changeAddress() on GMU.
----- Push Log Entry: 8/9/2002 10:30:05 AM
Received [True] from GMU.
----- Push Log Entry: 8/9/2002 10:30:05 AM
SUCCESS! (Pushed AddressLogID [11131] to GMU!
----- Push Log Entry: 8/9/2002 10:30:05 AM

```

```
Marked AddressEventID [10056] as completed.
----- Push Log Entry: 8/9/2002 10:30:06 AM
AddressID: [000005361 ] Target: [VRS]
----- Push Log Entry: 8/9/2002 10:30:07 AM
AddressID: [000005361]
----- Push Log Entry: 8/9/2002 10:30:07 AM
AddressType: [Business]
----- Push Log Entry: 8/9/2002 10:30:07 AM
AddressName: [Mr. James Smith]
----- Push Log Entry: 8/9/2002 10:30:07 AM
Address1: [3668 Apple Pie Ridge Road]
----- Push Log Entry: 8/9/2002 10:30:07 AM
Address2: [ ]
----- Push Log Entry: 8/9/2002 10:30:07 AM
City: [Winchester]
----- Push Log Entry: 8/9/2002 10:30:07 AM
State: [VA]
----- Push Log Entry: 8/9/2002 10:30:07 AM
ZIP: [22603]
----- Push Log Entry: 8/9/2002 10:30:07 AM
UpdateDate: [8/8/2002 12:00:00 AM]
----- Push Log Entry: 8/9/2002 10:30:07 AM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 10:30:07 AM
Calling changeAddress() on VRS.
----- Push Log Entry: 8/9/2002 10:30:12 AM
Received [False] from VRS.
----- Push Log Entry: 8/9/2002 10:30:12 AM
FAILURE! (Failed to push AddressLogID [11131] to VRS!
----- Push Log Entry: 8/9/2002 10:30:12 AM
AddressID: [000005361 ] Target: [VABC]
----- Push Log Entry: 8/9/2002 10:30:13 AM
AddressID: [000005361]
----- Push Log Entry: 8/9/2002 10:30:13 AM
AddressType: [Business]
----- Push Log Entry: 8/9/2002 10:30:13 AM
AddressName: [Mr. James Smith]
----- Push Log Entry: 8/9/2002 10:30:13 AM
Address1: [3668 Apple Pie Ridge Road]
----- Push Log Entry: 8/9/2002 10:30:13 AM
Address2: [ ]
----- Push Log Entry: 8/9/2002 10:30:13 AM
City: [Winchester]
----- Push Log Entry: 8/9/2002 10:30:13 AM
State: [VA]
----- Push Log Entry: 8/9/2002 10:30:13 AM
ZIP: [22603]
----- Push Log Entry: 8/9/2002 10:30:13 AM
UpdateDate: [8/8/2002 12:00:00 AM]
----- Push Log Entry: 8/9/2002 10:30:13 AM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 10:30:13 AM
Calling changeAddress() on VABC.
----- Push Log Entry: 8/9/2002 10:30:14 AM
Received [False] from VABC.
----- Push Log Entry: 8/9/2002 10:30:14 AM
FAILURE! (Failed to push AddressLogID [11131] to VABC!
----- Push Log Entry: 8/9/2002 10:30:14 AM
AddressID: [000005361 ] Target: [DHRM]
----- Push Log Entry: 8/9/2002 10:30:15 AM
AddressID: [000005361]
----- Push Log Entry: 8/9/2002 10:30:15 AM
AddressType: [Business]
----- Push Log Entry: 8/9/2002 10:30:15 AM
AddressName: [Mr. James Smith]
----- Push Log Entry: 8/9/2002 10:30:15 AM
Address1: [3668 Apple Pie Ridge Road]
----- Push Log Entry: 8/9/2002 10:30:15 AM
Address2: [ ]
----- Push Log Entry: 8/9/2002 10:30:15 AM
City: [Winchester]
```

```
----- Push Log Entry: 8/9/2002 10:30:15 AM
State: [VA]
----- Push Log Entry: 8/9/2002 10:30:15 AM
ZIP: [22603]
----- Push Log Entry: 8/9/2002 10:30:15 AM
UpdateDate: [8/8/2002 12:00:00 AM]
----- Push Log Entry: 8/9/2002 10:30:15 AM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 10:30:15 AM
Calling changeAddress() on DHRM.
----- Push Log Entry: 8/9/2002 10:30:26 AM
Received [True] from DHRM.
----- Push Log Entry: 8/9/2002 10:30:26 AM
SUCCESS! (Pushed AddressLogID [11131] to DHRM!
----- Push Log Entry: 8/9/2002 10:30:26 AM
Marked AddressEventID [10059] as completed.
----- DMVAddressPush : 8/9/2002 3:00:01 PM
----- Push Log Entry: 8/9/2002 3:00:01 PM
AddressID: [000005361 ] Target: [VRS]
----- Push Log Entry: 8/9/2002 3:00:03 PM
AddressID: [000005361]
----- Push Log Entry: 8/9/2002 3:00:03 PM
AddressType: [Business]
----- Push Log Entry: 8/9/2002 3:00:03 PM
AddressName: [Mr. James Smith]
----- Push Log Entry: 8/9/2002 3:00:03 PM
Address1: [3668 Apple Pie Ridge Road]
----- Push Log Entry: 8/9/2002 3:00:03 PM
Address2: [ ]
----- Push Log Entry: 8/9/2002 3:00:03 PM
City: [Winchester]
----- Push Log Entry: 8/9/2002 3:00:03 PM
State: [VA]
----- Push Log Entry: 8/9/2002 3:00:03 PM
ZIP: [22603]
----- Push Log Entry: 8/9/2002 3:00:03 PM
UpdateDate: [8/8/2002 12:00:00 AM]
----- Push Log Entry: 8/9/2002 3:00:03 PM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 3:00:03 PM
Calling changeAddress() on VRS.
----- Push Log Entry: 8/9/2002 3:00:09 PM
Received [True] from VRS.
----- Push Log Entry: 8/9/2002 3:00:09 PM
SUCCESS! (Pushed AddressLogID [11131] to VRS!
----- Push Log Entry: 8/9/2002 3:00:10 PM
Marked AddressEventID [10057] as completed.
----- Push Log Entry: 8/9/2002 3:00:10 PM
AddressID: [000005361 ] Target: [VABC]
----- Push Log Entry: 8/9/2002 3:00:11 PM
AddressID: [000005361]
----- Push Log Entry: 8/9/2002 3:00:11 PM
AddressType: [Business]
----- Push Log Entry: 8/9/2002 3:00:11 PM
AddressName: [Mr. James Smith]
----- Push Log Entry: 8/9/2002 3:00:11 PM
Address1: [3668 Apple Pie Ridge Road]
----- Push Log Entry: 8/9/2002 3:00:11 PM
Address2: [ ]
----- Push Log Entry: 8/9/2002 3:00:11 PM
City: [Winchester]
----- Push Log Entry: 8/9/2002 3:00:11 PM
State: [VA]
----- Push Log Entry: 8/9/2002 3:00:11 PM
ZIP: [22603]
----- Push Log Entry: 8/9/2002 3:00:11 PM
UpdateDate: [8/8/2002 12:00:00 AM]
----- Push Log Entry: 8/9/2002 3:00:11 PM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 3:00:11 PM
Calling changeAddress() on VABC.
```



```
----- Push Log Entry: 8/9/2002 3:00:11 PM
Received [False] from VABC.
----- Push Log Entry: 8/9/2002 3:00:11 PM
FAILURE! (Failed to push AddressLogID [11131] to VABC!
----- DMVAddressPush : 8/9/2002 3:15:02 PM
----- Push Log Entry: 8/9/2002 3:15:02 PM
AddressID: [000005361 ] Target: [VABC]
----- Push Log Entry: 8/9/2002 3:15:03 PM
AddressID: [000005361]
----- Push Log Entry: 8/9/2002 3:15:03 PM
AddressType: [Business]
----- Push Log Entry: 8/9/2002 3:15:03 PM
AddressName: [Mr. James Smith]
----- Push Log Entry: 8/9/2002 3:15:03 PM
Address1: [3668 Apple Pie Ridge Road]
----- Push Log Entry: 8/9/2002 3:15:03 PM
Address2: [ ]
----- Push Log Entry: 8/9/2002 3:15:03 PM
City: [Winchester]
----- Push Log Entry: 8/9/2002 3:15:03 PM
State: [VA]
----- Push Log Entry: 8/9/2002 3:15:03 PM
ZIP: [22603]
----- Push Log Entry: 8/9/2002 3:15:03 PM
UpdateDate: [8/8/2002 12:00:00 AM]
----- Push Log Entry: 8/9/2002 3:15:03 PM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 3:15:03 PM
Calling changeAddress() on VABC.
----- Push Log Entry: 8/9/2002 3:15:04 PM
Received [False] from VABC.
----- Push Log Entry: 8/9/2002 3:15:04 PM
FAILURE! (Failed to push AddressLogID [11131] to VABC!
----- DMVAddressPush : 8/9/2002 3:30:02 PM
----- Push Log Entry: 8/9/2002 3:30:02 PM
AddressID: [000005361 ] Target: [VABC]
----- Push Log Entry: 8/9/2002 3:30:03 PM
AddressID: [000005361]
----- Push Log Entry: 8/9/2002 3:30:03 PM
AddressType: [Business]
----- Push Log Entry: 8/9/2002 3:30:03 PM
AddressName: [Mr. James Smith]
----- Push Log Entry: 8/9/2002 3:30:03 PM
Address1: [3668 Apple Pie Ridge Road]
----- Push Log Entry: 8/9/2002 3:30:03 PM
Address2: [ ]
----- Push Log Entry: 8/9/2002 3:30:03 PM
City: [Winchester]
----- Push Log Entry: 8/9/2002 3:30:03 PM
State: [VA]
----- Push Log Entry: 8/9/2002 3:30:03 PM
ZIP: [22603]
----- Push Log Entry: 8/9/2002 3:30:03 PM
UpdateDate: [8/8/2002 12:00:00 AM]
----- Push Log Entry: 8/9/2002 3:30:03 PM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 3:30:03 PM
Calling changeAddress() on VABC.
----- Push Log Entry: 8/9/2002 3:30:04 PM
Received [False] from VABC.
----- Push Log Entry: 8/9/2002 3:30:04 PM
FAILURE! (Failed to push AddressLogID [11131] to VABC!
----- DMVAddressPush : 8/9/2002 3:45:02 PM
----- Push Log Entry: 8/9/2002 3:45:02 PM
AddressID: [000005361 ] Target: [VABC]
----- Push Log Entry: 8/9/2002 3:45:03 PM
AddressID: [000005361]
----- Push Log Entry: 8/9/2002 3:45:03 PM
AddressType: [Business]
----- Push Log Entry: 8/9/2002 3:45:03 PM
AddressName: [Mr. James Smith]
```

```
----- Push Log Entry: 8/9/2002 3:45:03 PM
Address1: [3668 Apple Pie Ridge Road]
----- Push Log Entry: 8/9/2002 3:45:03 PM
Address2: []
----- Push Log Entry: 8/9/2002 3:45:03 PM
City: [Winchester]
----- Push Log Entry: 8/9/2002 3:45:03 PM
State: [VA]
----- Push Log Entry: 8/9/2002 3:45:03 PM
ZIP: [22603]
----- Push Log Entry: 8/9/2002 3:45:03 PM
UpdateDate: [8/8/2002 12:00:00 AM]
----- Push Log Entry: 8/9/2002 3:45:03 PM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 3:45:03 PM
Calling changeAddress() on VABC.
----- Push Log Entry: 8/9/2002 3:45:04 PM
Received [False] from VABC.
----- Push Log Entry: 8/9/2002 3:45:04 PM
FAILURE! (Failed to push AddressLogID [11131] to VABC!
----- DMVAddressPush : 8/9/2002 4:00:02 PM
----- Push Log Entry: 8/9/2002 4:00:02 PM
AddressID: [000005361 ] Target: [VABC]
----- Push Log Entry: 8/9/2002 4:00:03 PM
AddressID: [000005361]
----- Push Log Entry: 8/9/2002 4:00:03 PM
AddressType: [Business]
----- Push Log Entry: 8/9/2002 4:00:03 PM
AddressName: [Mr. James Smith]
----- Push Log Entry: 8/9/2002 4:00:03 PM
Address1: [3668 Apple Pie Ridge Road]
----- Push Log Entry: 8/9/2002 4:00:03 PM
Address2: []
----- Push Log Entry: 8/9/2002 4:00:03 PM
City: [Winchester]
----- Push Log Entry: 8/9/2002 4:00:03 PM
State: [VA]
----- Push Log Entry: 8/9/2002 4:00:03 PM
ZIP: [22603]
----- Push Log Entry: 8/9/2002 4:00:03 PM
UpdateDate: [8/8/2002 12:00:00 AM]
----- Push Log Entry: 8/9/2002 4:00:03 PM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 4:00:03 PM
Calling changeAddress() on VABC.
----- Push Log Entry: 8/9/2002 4:00:04 PM
Received [False] from VABC.
----- Push Log Entry: 8/9/2002 4:00:04 PM
FAILURE! (Failed to push AddressLogID [11131] to VABC!
----- DMVAddressPush : 8/9/2002 4:06:58 PM
----- Push Log Entry: 8/9/2002 4:06:58 PM
AddressID: [000005361 ] Target: [VABC]
----- DMVAddressPush : 8/9/2002 4:15:02 PM
----- Push Log Entry: 8/9/2002 4:15:02 PM
AddressID: [000005361 ] Target: [VABC]
----- Push Log Entry: 8/9/2002 4:15:03 PM
AddressID: [000005361]
----- Push Log Entry: 8/9/2002 4:15:03 PM
AddressType: [Business]
----- Push Log Entry: 8/9/2002 4:15:03 PM
AddressName: [Mr. James Smith]
----- Push Log Entry: 8/9/2002 4:15:03 PM
Address1: [3668 Apple Pie Ridge Road]
----- Push Log Entry: 8/9/2002 4:15:03 PM
Address2: []
----- Push Log Entry: 8/9/2002 4:15:03 PM
City: [Winchester]
----- Push Log Entry: 8/9/2002 4:15:03 PM
State: [VA]
----- Push Log Entry: 8/9/2002 4:15:03 PM
ZIP: [22603]
```

```
----- Push Log Entry: 8/9/2002 4:15:03 PM
UpdateDate: [8/8/2002 12:00:00 AM]
----- Push Log Entry: 8/9/2002 4:15:03 PM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 4:15:03 PM
Calling changeAddress() on VABC.
----- Push Log Entry: 8/9/2002 4:15:04 PM
Received [False] from VABC.
----- Push Log Entry: 8/9/2002 4:15:04 PM
FAILURE! (Failed to push AddressLogID [11131] to VABC!
----- DMVAddressPush : 8/9/2002 4:30:02 PM
----- Push Log Entry: 8/9/2002 4:30:02 PM
AddressID: [000005361 ] Target: [VABC]
----- Push Log Entry: 8/9/2002 4:30:03 PM
AddressID: [000005361]
----- Push Log Entry: 8/9/2002 4:30:03 PM
AddressType: [Business]
----- Push Log Entry: 8/9/2002 4:30:03 PM
AddressName: [Mr. James Smith]
----- Push Log Entry: 8/9/2002 4:30:03 PM
Address1: [3668 Apple Pie Ridge Road]
----- Push Log Entry: 8/9/2002 4:30:03 PM
Address2: [ ]
----- Push Log Entry: 8/9/2002 4:30:03 PM
City: [Winchester]
----- Push Log Entry: 8/9/2002 4:30:03 PM
State: [VA]
----- Push Log Entry: 8/9/2002 4:30:03 PM
ZIP: [22603]
----- Push Log Entry: 8/9/2002 4:30:03 PM
UpdateDate: [8/8/2002 12:00:00 AM]
----- Push Log Entry: 8/9/2002 4:30:03 PM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 4:30:03 PM
Calling changeAddress() on VABC.
----- Push Log Entry: 8/9/2002 4:30:04 PM
Received [False] from VABC.
----- Push Log Entry: 8/9/2002 4:30:04 PM
FAILURE! (Failed to push AddressLogID [11131] to VABC!
----- DMVAddressPush : 8/9/2002 4:45:02 PM
----- Push Log Entry: 8/9/2002 4:45:02 PM
AddressID: [000005361 ] Target: [VABC]
----- Push Log Entry: 8/9/2002 4:45:03 PM
AddressID: [000005361]
----- Push Log Entry: 8/9/2002 4:45:03 PM
AddressType: [Business]
----- Push Log Entry: 8/9/2002 4:45:03 PM
AddressName: [Mr. James Smith]
----- Push Log Entry: 8/9/2002 4:45:03 PM
Address1: [3668 Apple Pie Ridge Road]
----- Push Log Entry: 8/9/2002 4:45:03 PM
Address2: [ ]
----- Push Log Entry: 8/9/2002 4:45:03 PM
City: [Winchester]
----- Push Log Entry: 8/9/2002 4:45:03 PM
State: [VA]
----- Push Log Entry: 8/9/2002 4:45:03 PM
ZIP: [22603]
----- Push Log Entry: 8/9/2002 4:45:03 PM
UpdateDate: [8/8/2002 12:00:00 AM]
----- Push Log Entry: 8/9/2002 4:45:03 PM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 4:45:03 PM
Calling changeAddress() on VABC.
----- Push Log Entry: 8/9/2002 4:45:04 PM
Received [False] from VABC.
----- Push Log Entry: 8/9/2002 4:45:04 PM
FAILURE! (Failed to push AddressLogID [11131] to VABC!
----- DMVAddressPush : 8/9/2002 5:00:02 PM
----- Push Log Entry: 8/9/2002 5:00:02 PM
AddressID: [000005361 ] Target: [VABC]
```

```
----- Push Log Entry: 8/9/2002 5:00:03 PM
AddressID: [000005361]
----- Push Log Entry: 8/9/2002 5:00:03 PM
AddressType: [Business]
----- Push Log Entry: 8/9/2002 5:00:03 PM
AddressName: [Mr. James Smith]
----- Push Log Entry: 8/9/2002 5:00:03 PM
Address1: [3668 Apple Pie Ridge Road]
----- Push Log Entry: 8/9/2002 5:00:03 PM
Address2: []
----- Push Log Entry: 8/9/2002 5:00:03 PM
City: [Winchester]
----- Push Log Entry: 8/9/2002 5:00:03 PM
State: [VA]
----- Push Log Entry: 8/9/2002 5:00:03 PM
ZIP: [22603]
----- Push Log Entry: 8/9/2002 5:00:03 PM
UpdateDate: [8/8/2002 12:00:00 AM]
----- Push Log Entry: 8/9/2002 5:00:03 PM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 5:00:03 PM
Calling changeAddress() on VABC.
----- Push Log Entry: 8/9/2002 5:00:04 PM
Received [False] from VABC.
----- Push Log Entry: 8/9/2002 5:00:04 PM
FAILURE! (Failed to push AddressLogID [11131] to VABC!
----- DMVAddressPush : 8/9/2002 5:15:02 PM
----- Push Log Entry: 8/9/2002 5:15:02 PM
AddressID: [000005361 ] Target: [VABC]
----- Push Log Entry: 8/9/2002 5:15:03 PM
AddressID: [000005361]
----- Push Log Entry: 8/9/2002 5:15:03 PM
AddressType: [Business]
----- Push Log Entry: 8/9/2002 5:15:03 PM
AddressName: [Mr. James Smith]
----- Push Log Entry: 8/9/2002 5:15:03 PM
Address1: [3668 Apple Pie Ridge Road]
----- Push Log Entry: 8/9/2002 5:15:03 PM
Address2: []
----- Push Log Entry: 8/9/2002 5:15:03 PM
City: [Winchester]
----- Push Log Entry: 8/9/2002 5:15:03 PM
State: [VA]
----- Push Log Entry: 8/9/2002 5:15:03 PM
ZIP: [22603]
----- Push Log Entry: 8/9/2002 5:15:03 PM
UpdateDate: [8/8/2002 12:00:00 AM]
----- Push Log Entry: 8/9/2002 5:15:03 PM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 5:15:03 PM
Calling changeAddress() on VABC.
----- Push Log Entry: 8/9/2002 5:15:04 PM
Received [False] from VABC.
----- Push Log Entry: 8/9/2002 5:15:04 PM
FAILURE! (Failed to push AddressLogID [11131] to VABC!
----- Push Log Entry: 8/9/2002 5:15:04 PM
AddressID: [001000070 ] Target: [GMU]
----- Push Log Entry: 8/9/2002 5:15:05 PM
AddressID: [001000070]
----- Push Log Entry: 8/9/2002 5:15:05 PM
AddressType: [Home/Street]
----- Push Log Entry: 8/9/2002 5:15:05 PM
AddressName: [Mr. Larry A. Spear]
----- Push Log Entry: 8/9/2002 5:15:05 PM
Address1: [100 University Drive]
----- Push Log Entry: 8/9/2002 5:15:05 PM
Address2: [Your Street]
----- Push Log Entry: 8/9/2002 5:15:05 PM
City: [Fairfax]
----- Push Log Entry: 8/9/2002 5:15:05 PM
State: [VA]
```

```
----- Push Log Entry: 8/9/2002 5:15:05 PM
ZIP: [22032]
----- Push Log Entry: 8/9/2002 5:15:05 PM
UpdateDate: [8/9/2002 6:06:49 PM]
----- Push Log Entry: 8/9/2002 5:15:05 PM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 5:15:05 PM
Calling changeAddress() on GMU.
----- Push Log Entry: 8/9/2002 5:15:06 PM
Received [True] from GMU.
----- Push Log Entry: 8/9/2002 5:15:06 PM
SUCCESS! (Pushed AddressLogID [11135] to GMU!
----- Push Log Entry: 8/9/2002 5:15:06 PM
Marked AddressEventID [10060] as completed.
----- Push Log Entry: 8/9/2002 5:15:06 PM
AddressID: [001000070 ] Target: [VRS]
----- Push Log Entry: 8/9/2002 5:15:07 PM
AddressID: [001000070]
----- Push Log Entry: 8/9/2002 5:15:07 PM
AddressType: [Home/Street]
----- Push Log Entry: 8/9/2002 5:15:07 PM
AddressName: [Mr. Larry A. Spear]
----- Push Log Entry: 8/9/2002 5:15:07 PM
Address1: [100 University Drive]
----- Push Log Entry: 8/9/2002 5:15:07 PM
Address2: [Your Street]
----- Push Log Entry: 8/9/2002 5:15:07 PM
City: [Fairfax]
----- Push Log Entry: 8/9/2002 5:15:07 PM
State: [VA]
----- Push Log Entry: 8/9/2002 5:15:07 PM
ZIP: [22032]
----- Push Log Entry: 8/9/2002 5:15:07 PM
UpdateDate: [8/9/2002 6:06:49 PM]
----- Push Log Entry: 8/9/2002 5:15:07 PM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 5:15:07 PM
Calling changeAddress() on VRS.
----- Push Log Entry: 8/9/2002 5:15:18 PM
Received [True] from VRS.
----- Push Log Entry: 8/9/2002 5:15:18 PM
SUCCESS! (Pushed AddressLogID [11135] to VRS!
----- Push Log Entry: 8/9/2002 5:15:19 PM
Marked AddressEventID [10061] as completed.
----- Push Log Entry: 8/9/2002 5:15:19 PM
AddressID: [001000070 ] Target: [VABC]
----- Push Log Entry: 8/9/2002 5:15:19 PM
AddressID: [001000070]
----- Push Log Entry: 8/9/2002 5:15:19 PM
AddressType: [Home/Street]
----- Push Log Entry: 8/9/2002 5:15:19 PM
AddressName: [Mr. Larry A. Spear]
----- Push Log Entry: 8/9/2002 5:15:19 PM
Address1: [100 University Drive]
----- Push Log Entry: 8/9/2002 5:15:19 PM
Address2: [Your Street]
----- Push Log Entry: 8/9/2002 5:15:19 PM
City: [Fairfax]
----- Push Log Entry: 8/9/2002 5:15:19 PM
State: [VA]
----- Push Log Entry: 8/9/2002 5:15:19 PM
ZIP: [22032]
----- Push Log Entry: 8/9/2002 5:15:19 PM
UpdateDate: [8/9/2002 6:06:49 PM]
----- Push Log Entry: 8/9/2002 5:15:19 PM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 5:15:19 PM
Calling changeAddress() on VABC.
----- Push Log Entry: 8/9/2002 5:15:19 PM
Received [True] from VABC.
----- Push Log Entry: 8/9/2002 5:15:19 PM
```

```
SUCCESS! (Pushed AddressLogID [11135] to VABC!
----- Push Log Entry: 8/9/2002 5:15:19 PM
Marked AddressEventID [10062] as completed.
----- Push Log Entry: 8/9/2002 5:15:19 PM
AddressID: [001000070 ] Target: [DHRM]
----- Push Log Entry: 8/9/2002 5:15:20 PM
AddressID: [001000070]
----- Push Log Entry: 8/9/2002 5:15:20 PM
AddressType: [Home/Street]
----- Push Log Entry: 8/9/2002 5:15:20 PM
AddressName: [Mr. Larry A. Spear]
----- Push Log Entry: 8/9/2002 5:15:20 PM
Address1: [100 University Drive]
----- Push Log Entry: 8/9/2002 5:15:20 PM
Address2: [Your Street]
----- Push Log Entry: 8/9/2002 5:15:20 PM
City: [Fairfax]
----- Push Log Entry: 8/9/2002 5:15:20 PM
State: [VA]
----- Push Log Entry: 8/9/2002 5:15:20 PM
ZIP: [22032]
----- Push Log Entry: 8/9/2002 5:15:20 PM
UpdateDate: [8/9/2002 6:06:49 PM]
----- Push Log Entry: 8/9/2002 5:15:20 PM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 5:15:20 PM
Calling changeAddress() on DHRM.
----- Push Log Entry: 8/9/2002 5:15:30 PM
Received [True] from DHRM.
----- Push Log Entry: 8/9/2002 5:15:30 PM
SUCCESS! (Pushed AddressLogID [11135] to DHRM!
----- Push Log Entry: 8/9/2002 5:15:30 PM
Marked AddressEventID [10063] as completed.
----- DMVAddressPush : 8/9/2002 5:30:02 PM
----- Push Log Entry: 8/9/2002 5:30:02 PM
AddressID: [000005361 ] Target: [VABC]
----- Push Log Entry: 8/9/2002 5:30:03 PM
AddressID: [000005361]
----- Push Log Entry: 8/9/2002 5:30:03 PM
AddressType: [Business]
----- Push Log Entry: 8/9/2002 5:30:03 PM
AddressName: [Mr. James Smith]
----- Push Log Entry: 8/9/2002 5:30:03 PM
Address1: [3668 Apple Pie Ridge Road]
----- Push Log Entry: 8/9/2002 5:30:03 PM
Address2: [ ]
----- Push Log Entry: 8/9/2002 5:30:03 PM
City: [Winchester]
----- Push Log Entry: 8/9/2002 5:30:03 PM
State: [VA]
----- Push Log Entry: 8/9/2002 5:30:03 PM
ZIP: [22603]
----- Push Log Entry: 8/9/2002 5:30:03 PM
UpdateDate: [8/8/2002 12:00:00 AM]
----- Push Log Entry: 8/9/2002 5:30:03 PM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 5:30:03 PM
Calling changeAddress() on VABC.
----- Push Log Entry: 8/9/2002 5:30:04 PM
Received [False] from VABC.
----- Push Log Entry: 8/9/2002 5:30:04 PM
FAILURE! (Failed to push AddressLogID [11131] to VABC!
----- DMVAddressPush : 8/9/2002 5:45:02 PM
----- Push Log Entry: 8/9/2002 5:45:02 PM
AddressID: [000005361 ] Target: [VABC]
----- Push Log Entry: 8/9/2002 5:45:03 PM
AddressID: [000005361]
----- Push Log Entry: 8/9/2002 5:45:03 PM
AddressType: [Business]
----- Push Log Entry: 8/9/2002 5:45:03 PM
AddressName: [Mr. James Smith]
```

```
----- Push Log Entry: 8/9/2002 5:45:03 PM
Address1: [3668 Apple Pie Ridge Road]
----- Push Log Entry: 8/9/2002 5:45:03 PM
Address2: [ ]
----- Push Log Entry: 8/9/2002 5:45:03 PM
City: [Winchester]
----- Push Log Entry: 8/9/2002 5:45:03 PM
State: [VA]
----- Push Log Entry: 8/9/2002 5:45:03 PM
ZIP: [22603]
----- Push Log Entry: 8/9/2002 5:45:03 PM
UpdateDate: [8/8/2002 12:00:00 AM]
----- Push Log Entry: 8/9/2002 5:45:03 PM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 5:45:03 PM
Calling changeAddress() on VABC.
----- Push Log Entry: 8/9/2002 5:45:04 PM
Received [False] from VABC.
----- Push Log Entry: 8/9/2002 5:45:04 PM
FAILURE! (Failed to push AddressLogID [11131] to VABC!
----- DMVAddressPush : 8/9/2002 6:00:02 PM
----- Push Log Entry: 8/9/2002 6:00:02 PM
AddressID: [000005361 ] Target: [VABC]
----- Push Log Entry: 8/9/2002 6:00:03 PM
AddressID: [000005361]
----- Push Log Entry: 8/9/2002 6:00:03 PM
AddressType: [Business]
----- Push Log Entry: 8/9/2002 6:00:03 PM
AddressName: [Mr. James Smith]
----- Push Log Entry: 8/9/2002 6:00:03 PM
Address1: [3668 Apple Pie Ridge Road]
----- Push Log Entry: 8/9/2002 6:00:03 PM
Address2: [ ]
----- Push Log Entry: 8/9/2002 6:00:03 PM
City: [Winchester]
----- Push Log Entry: 8/9/2002 6:00:03 PM
State: [VA]
----- Push Log Entry: 8/9/2002 6:00:03 PM
ZIP: [22603]
----- Push Log Entry: 8/9/2002 6:00:03 PM
UpdateDate: [8/8/2002 12:00:00 AM]
----- Push Log Entry: 8/9/2002 6:00:03 PM
Originator: [DMV]
----- Push Log Entry: 8/9/2002 6:00:03 PM
Calling changeAddress() on VABC.
----- Push Log Entry: 8/9/2002 6:00:04 PM
Received [False] from VABC.
----- Push Log Entry: 8/9/2002 6:00:04 PM
FAILURE! (Failed to push AddressLogID [11131] to VABC!
----- DMVAddressPush : 8/9/2002 6:15:02 PM
----- Push Log Entry: 8/9/2002 6:15:02 PM
AddressID: [000005361 ] Target: [VABC]
----- Push Log Entry: 8/9/2002 6:15:03 PM
AddressID: [000005361]
----- Push Log Entry: 8/9/2002 6:15:03 PM
AddressType: [Business]
----- Push Log Entry: 8/9/2002 6:15:03 PM
AddressName: [Mr. James Smith]
----- Push Log Entry: 8/9/2002 6:15:03 PM
Address1: [3668 Apple Pie Ridge Road]
----- Push Log Entry: 8/9/2002 6:15:03 PM
Address2: [ ]
----- Push Log Entry: 8/9/2002 6:15:03 PM
City: [Winchester]
----- Push Log Entry: 8/9/2002 6:15:03 PM
State: [VA]
----- Push Log Entry: 8/9/2002 6:15:03 PM
ZIP: [22603]
----- Push Log Entry: 8/9/2002 6:15:03 PM
UpdateDate: [8/8/2002 12:00:00 AM]
----- Push Log Entry: 8/9/2002 6:15:03 PM
```

```
Originator: [DMV]
----- Push Log Entry: 8/9/2002 6:15:03 PM
Calling changeAddress() on VABC.
----- Push Log Entry: 8/9/2002 6:15:04 PM
Received [False] from VABC.
----- Push Log Entry: 8/9/2002 6:15:04 PM
FAILURE! (Failed to push AddressLogID [11131] to VABC!
```



## Appendix G: Team 2 Final Report – VABC & BEA

### 1. PROJECT PLAN (SCHEDULE)

Task – Assignment	» Time (days)	% Complete	Expected Completion Date
Write Business Scenarios	1	100%	6/25
Write Test Plan	1	100%	6/28
Become familiar with XML file / schema	1	100%	
Become familiar with SOAP	1 – 2	100%	6/28
Become familiar with UDDI	1	100%	6/28
Become familiar with JXM, JXRPC	3 – 4	N/A	6/28
Understand WSDL format	1	100%	6/28
Learn to use web services with WebLogic	1	20%	6/28
Meeting regarding business scenarios and BEA tools	1/2	100%	6/25
Setup development environment	1	100%	7/10
Use WSDL to write web services	1	100%	7/19
Generate web services structure using WSDL	1	100%	7/19
Implement methods / business logic	1	100%	7/19
Deploy web services in BEA Weblogic (app. server) and register services in UDDI	1	100%	7/19
Meeting to review logic and finalize implementation	0		TBD
Write a client to look up UDDI web services	1	100%	7/26
Use WSDL to invoke web services of other parties	1	100%	7/26
Implement methods / business logic	1	100%	7/26
Testing	3	100%	8/1
Meeting to review implementation of client side invocation of services and associated business logic	0		TBD
Prepare Presentations for Workgroup	1		8/1
Workgroup Meeting and Presentations	4		8/19
Prepare COTS Report	TBD		8/25
Workgroup of draft reports	TBD		9/9
Submission of final report to COTS	TBD		9/10

Table 1 Project Plan

### 2. LIST OF TEAM NAME, MEMBERS, ROLES AND RESPONSIBILITIES

**Team Name:**

Team 2 – VABC, BEA

**Team Members:**

Name	Organization	Email	Phone
Will Howery	BEA	<a href="mailto:whowery@bea.com">whowery@bea.com</a>	804.921.7433
Dan Lender	BEA	<a href="mailto:daniel.lender@bea.com">daniel.lender@bea.com</a>	571.382.2454
David Hassen	VABC	<a href="mailto:dwhassn@abc.state.va.us">dwhassn@abc.state.va.us</a>	804.213.4492
Sue Shirbacheh	VABC	<a href="mailto:sashirh@abc.state.va.us">sashirh@abc.state.va.us</a>	804.213.4725

Table 2 List of Team Members

### Roles and Responsibilities:

Dan Lender - Coordinate resources and supply BEA software necessary to implement the project.

Will Howery - Lead the technical and design efforts for implementing the web services using best practices and tools available from BEA. Participate in the prototype design and review development activities.

David Hassen - Coordinate any resources required of the Virginia ABC to meet the desired objectives. Participate in the design and development of the web services components and integrate with a Virginia ABC test application.

Sue Shirbacheh - Participate in the design and development of the web services components and integrate with a Virginia ABC test application.

### 3. PROOF-OF-CONCEPT DESIGN

This proof-of-concept outlines the basic interfaces and data representations needed to implement the address change and update Web Services project.

#### Terms:

Address type - refers to the types of address that is being updated, pushed, or registered for.

Address Identifier (ID) - The address identifier is a unique identifier for a given address. This ID allows correlation of an updated address from agency to agency and system to system. Without a defined address ID there would be duplication of addresses and names.

Entity type - refers to the individual or a business entity. The reason to delineate the type of entity is to determine the address ID. The presumption is that if the address entity is an individual then the social security number of that individual can be used as the identifier of the address. If the address entity is a business then the address identifier could be the

federal tax ID. The only businesses that do not have federal tax ids are sole proprietors, and then the social security number of the owner could be used.

Notification or Push - Agencies can select if they want addresses sent directly to them (push), or if they want to be notified of the address changed and then have the ability to go back and access the address at the time of their choosing.

### Class diagram of interfaces and data definitions:

The following class diagram details the interfaces for the subscription service and the data elements needed for an address representation and subscriber list.

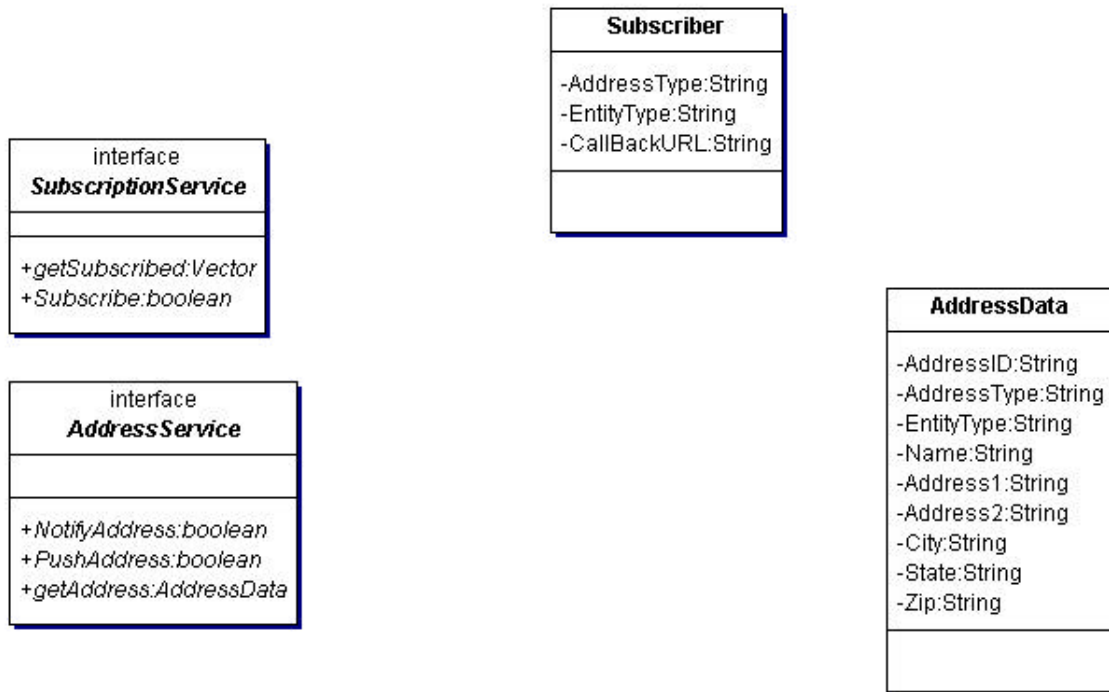


Figure 1 Address WebServices class diagram

The Subscriber class represents the information that will be stored by the subscription service, and subsequently returned in a list of subscribers when an agency invokes the getSubscriber service on the subscription service. The Address Data class represents the data and id data for an address.

### Suggested XML Schemas:

The following structure represents a sample generic subscriber list.

```

<subscriberlist>
  <subscriber addresstype='billing' entitytype='business'>
    http://whowery:7001/webService/NotifyAddress.wsdl
  </subscriber>
</subscriberlist>

```

```
<subscriber addresstype='billing' entitytype='business'>
  http://vabc:7001/webservice/PushAddress.wsdl
</subscriber>
</subscriberlist>
```

Figure 2 Suggested XML representation of a subscriber list

The following structure represents a sample generic address.

```
<address>
  <addresstype>billing</addresstype>
  <addressid>FEDIDXXXX</addressid>
  <timestamp>xsd:dateTime</timestamp>
  <source>VABC</source>
  <address1>Apt 123</address1>
  <address2>1259 Swallow Creek Rd</address2>
  <city>Richmond</city>
  <state>VA</state>
  <zip>23233</zip>
</address>
```

Figure 3 Suggested XML representation of an address

The following describes each address tag:

- <addresstype>
  - The type of address (e.g, home, business, billing, garage, mailing)
  - Mandatory
- <addressid>
  - Identifier for an address (typically the user's SSN or a businesses tax id)
  - Number - 9 digits in length
  - Mandatory
- <timestamp>
  - Time that the address was last updated (used in the log and query results)
  - xsd:dateTime
  - Optional
- <source>
  - Source URL of address change (used in the log and query results)
  - Optional
- <address1>
  - First line of an address
  - Alphanumeric
  - Mandatory
- <address2>
  - Second line of an address
  - Alphanumeric
  - Optional
- <city>

- City name
  - Alphanumeric
  - Mandatory
- <state>
  - State abbreviation
  - 2 characters
  - Mandatory
- <zip>
  - Zip code
  - Number – 5 digits in length
  - Mandatory
  - Optionally could contain 9 digit zip code with a hyphen. NNNNN-NNNN

When an address XML document is received by an agency that document will be handled in the most appropriate mechanism for that agency. The agency will also keep an operation log of the document. This log will fulfill two purposes: 1) it will provide the bases for sending back address changes to a queryAddress request and 2) it will provide a logging instance to test and validate that the service received all addresses sent to it. After the address change WebServices have been deployed a test can be run with all changes going to a log. This log can then be analyzed to validate that each agency had in fact propagated address changes and received any address changes propagated.

The following structure represents a sample generic address log.

```
<addresslog>
  <address>
    <addresstype>billing</addresstype>
    <addressid>FEDIDXXX</addressid>
  <timestamp>xsd:dateTime</timestamp>
    <source>VABC</source>
    <address1>Apt 123</address1>
    <address2>1259 Swallow Creek Rd</address2>
    <city>Richmond</city>
    <state>VA</state>
    <zip>23233</zip>
  </address>
  <address>
    .
    .
    .
  </address>
</addresslog>
```

Figure 4 Suggested XML representation of an address log

The addresslog schema example will contain multiple address entries. Reusing the address structure will reduce the effort in managing the log entries. The only difference in the address log entry from the address schema will be the mandatory addition of a source and a time stamp. The source will identify the url which sent this message to the receiving agency. The time stamp will be in the Year/Month/Day Hour:Minute:Second format as yyyy/mm/dd

hh:mm:ss. For purposes of the testing we can use Virginia local time. If the system were to be expanded we would want to record GMT or include a time zone identifier.

### WSDL document:

The following WSDL implementation will represent the interfaces discussed in the workshop meeting. As follows:

```
public interface AddressChangeWebService {  
    public AddressData getAddress(String addressID, String addressType)  
        throws InvalidAddressID;  
    public boolean changeAddress(AddressData aAddressData, String  
        OriginatorURL);  
    public AddressLog queryAddress(Date startDate, Date endDate, String  
        addressType, String OriginatorURL);  
}
```

The WSDL document defines the AddressData XML structure. The AddressLog coming from the queryAddress still needs some work to not be more portable.

See Appendix 1 for WSDL document.

### UDDI Registration:

Agencies and the subscription service will register in a UDDI registry. This registry will contain information about the services exposed and a Web Services Description Language (WSDL) document that contains the necessary details for other agencies to access and invoke the various address web services.

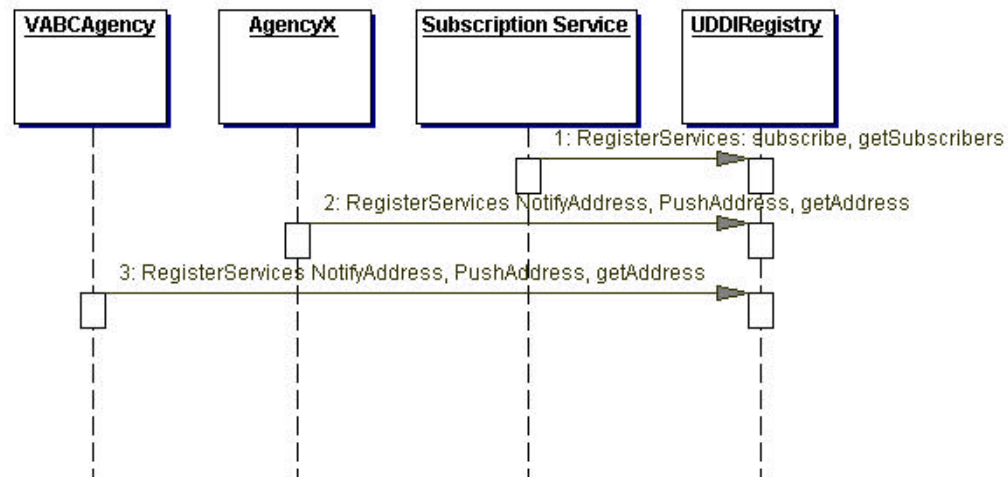


Figure 5 Agencies registering Web Services with the UDDI

### Subscription Services:

An agency subscribing for address changes is the first step in Address Web Services integration. An agency subscribes with the address subscription service. The agency supplies the type of address it wants to be updated with, the manner of being updated (e.g. notification or push), the type of entity (e.g. individual or company) and a call back URL for the WSDL that will provide the update service. The subscription service will keep this information and return the subscriber list to any other service that request the supplied address type and entity type.

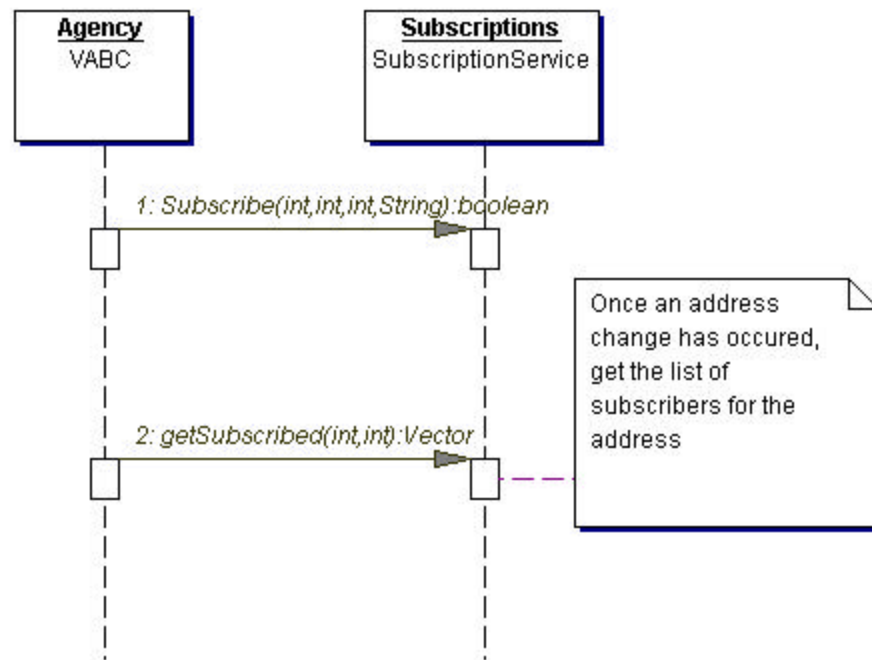


Figure 6 Subscribing to the subscription service, and getting the list of subscribers

**Signature for subscribing:**

```
boolean Subscribe(String addressType, String entityType, String  
updateType, String callBackURL);
```

For the second subscription service, An agency has had an address change and now wants to ask the subscription service for the list of agencies that have subscribed to updates for this type of address change. The agency will send the address type, and entity type to the `getSubscription` service and the subscription service will return a list of subscribers with the mode of update and callback URLs for the services that will be used to update the subscribing agencies.

**Signature for getSubscribed:**

```
Vector getSubscribed(String addressType, String entityType);
```

## Processing a Change In Address:

The following scenario details the actions that will occur when an address is changed.

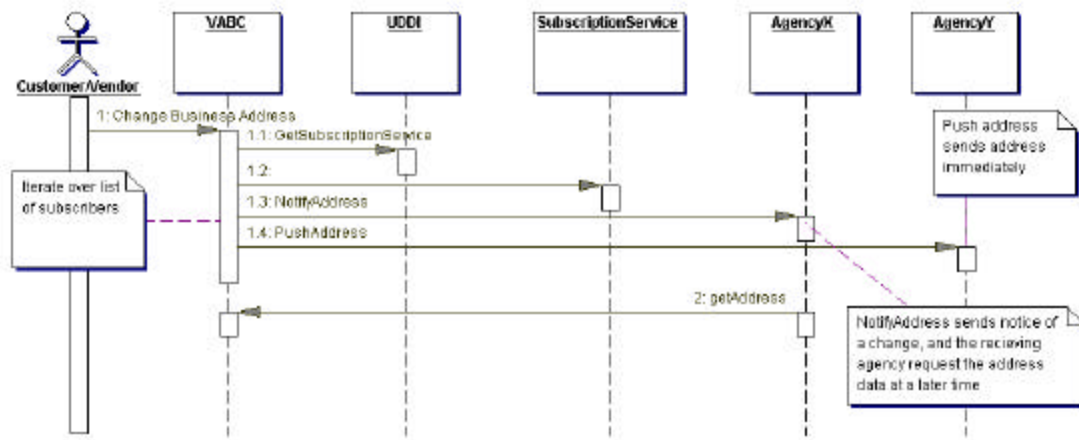


Figure 7 Address change scenario with NotifyAddress, PushAddress and getAddress.

When a customer or Vendor changes an address with a host agency that agency will start the process of sending updates or notifications of address changes to other agencies that have subscribed for that type of address change. The first step for the host agency will be to check in the UDDI and validate the subscriber service. Then the host agency will invoke the getSubscriber service on the subscription WebService. The getSubscriber service will pass in the address type, and entity type information. From this information the subscription service will search for subscribers who have subscribed for that address type with that entity type. The subscription service will create a list of subscribers (note Fig 1; for the Subscriber object content) this list will be returned to the host service. When returned the internal object list will be serialized to an XML document for the host service to consume and process.

### Signature of the getSubscriber service:

```
Vector getSubscribed(String addressType, String entityType);
```

Once the host service has the list of subscribers the host service will iterate through the list and either notify or push the address change to the subscribed agencies. When a Notify address is invoked the host service will send the address type entity type, address id and a call back URL for the subscribing agency to use when later requesting the address data from the host agency.

### Signature of NotifyAddress:

```
boolean NotifyAddress(String addressType, String entityType, String  
addressID, String CallbackURL);
```

When a push address is requested from the subscribed agency, the host agency will immediately send the address data to the subscribing agency.



**Signature of the PushAddress:**

```
boolean PushAddress(String addressType, String entityType, String  
    addressID, AddressData addressData);
```

At some later time the agency that received the NotifyAddress will need to request that address information from the host agency. The purpose of the NotifyAddress services is to allow agencies to load balance their communications and selectively choose when they well receiving address data, and updating their systems. The subscribing agency will store the notification information and at a later time invoke a getAddress on the host agency to retrieve the actual address data.

**Signature of the getAddress service:**

```
AddressData getAddress(String addressType, String entityType, String  
    addressID);
```

This section represents the result of analyzing the data and signatures needed to implement the address change functionality across agencies. The section does not prescribe how to implement the security features needed for these services. The presumption is that the Web Services clients will authenticate them selves with the agency services servers and invoke the said services via HTTPS with SSL security.

#### **4. BUSINESS/FUNCTIONAL REQUIREMENTS**

**Service Requirements:**

- The system will provide a subscription services that will take information pertaining to a type of address, a form of update mechanism (i.e. push or notify), and a callback URL (wsdl).
- The system will get subscribers, input address type, list of update type (push or notify) and a callback URL (wsdl).

**General Requirements:**

- The system must invoke to subscribe noting the address type, update type, and callback URL.
- The system must get subscribers, their address type, a return list with the update mechanism (push or notify), and a call back URL.
- The system must invoke the callback URL (wsdl) for a notify.
- The system must invoke callback ULR (wsdl) for a push.
- The system must contain a method to get address type which will indicate if the type is an individual or business and a unique identifier.
- On notify, the system must indicate the type, individual business, and a unique identifier.
- On push, the system must indicate the type, individual or business, and the data.

**Functional/Technical Assumptions:**

- The system will assume all vendors/contacts have Virginia Addresses for the purpose of this prototype.
- The system will assume that a company address change will be applied to all contacts for the given company.
- Design will begin under the assumption that the address schemas that will be published will be determined by the group.

## **5. PROJECT EQUIPMENT DESCRIPTION**

### **Hardware:**

- Dell Laptop
- 1 GHZ Intel Processor
- 30 GB Hard Drive
- 512 MB Memory

### **Software:**

- Windows 2000
- BEA Weblogic Application Server
- BEA Workshop
- Pointbase Database

## **6. ACQUISITION AND INSTALLATION ACTIVITIES**

Additional hardware will not be required for this prototype. BEA has agreed to supply copies of the Weblogic application server and Workshop tool.

## **7. SERVER REQUIREMENTS**

The purchase of additional resources is not required.

## **8. NETWORKING REQUIREMENTS**

The server and entire application environment will run outside of the VABC firewall. This will require the system administrator to setup a unique IP address for this machine. No additional networking requirements are anticipated.

## **9. TRAINING REQUIREMENTS**

- Self Study – XML
- Self Study – SOAP
- Self Study – UDDI
- Self Study – JXM, JXRPC

- Self Study – WSDL
- BEA assisted – Weblogic / Web Services Integration
- BEA assisted – Workshop
- BEA assisted – technical guidance

## 10. TESTING PLAN

Please see the attached Test Plan document.

## 11. DEVELOPMENT/TECHNICAL

None at this time.

## 12. PROTOCOL/SPECIFICATION ISSUES

### **WSDL Interface design and implementation:**

The original workgroup WSDL design prescribed a single service with three operations calling the getAddress, ChangeAddress and queryAddress. Each operation in the interface either took an AddressData structure as a parameter or returned it as an object. In the query Address operation the AddressData structure was passed back as an Array. The design of this WSDL was purposefully picked to demonstrate the ability to handle two levels of data complexity. The first level was the AddressData object itself. Returning and converting a complex structure requires the various web services servers and clients to be able to convert their native object types to an XML structure and then convert that xml document back into an object structure on the client software. The second level of data complexity was to be able to take a collection of these complex objects and return them in the Web services prescribed Array xml schema data type.

The Original web service definition prescribed the following structure:

<b>WSDL</b>	<b>Service</b>	<b>Operation</b>
AddressChange	AddressChangeBean	AddressData getAddress(String addressID, String addressType) throws InvalidAddressID;
AddressChange	AddressChangeBean	boolean changeAddress(AddressData aAddressData);
AddressChange	AddressChangeBean	AddressData[] queryAddress(Date startDate, Date endDate, String addressType, String OriginatorURL);

Table 3 Interface “A”: The original definition

For explanation purposes let's call this original web service definition Interface “A”. As the workgroup progressed and the various agencies built their web services another WSDL definition was implemented by several of the agencies as follows:

WSDL	Service	Operation
GetAddress	getAddress	AddressData getAddress(String addressID, String addressType) throws InvalidAddressID;
ChangeAddress	changeAddress	boolean changeAddress(AddressData aAddressData);
QueryAddress	queryAddress	AddressData[] queryAddress(Date startDate, Date endDate, String addressType, String OriginatorURL);

Table 4 Interface “B”: Each operation separated into its own Service and WSDL.

While the Interface “B” pattern was functional it did have an impact on the final complexity required to implement clients to utilize the web services. The following list details the complexities introduced by Interface “B”:

- Multiple Soap Ports
- Multiple WSDLs
- Multiple Static Clients
- Broke the Interface contract

Multiple SOAP ports - Need to be defined and configured, and maintained for multiple services. Multiple anything if not required is a bad thing in an integration project. There are enough moving parts without introducing additional components.

Multiple WSDLs - Are adding complexity for both the client and the server. . Again this creates more moving parts, best practices encourage designs to use the “Keep it Simple” axiom.

Multiple Static Clients - For a static client, each WSDL must be consumed and produce its native code implementation to invoke and consume that web service. However, if the lookup into the UDDI with a dynamic invocation had been completely implemented the separate WSDLs for each operation would not have added complexity to the clients. But, since the state of WebServices at this time is static clients, this separate WSDL for each operation does add work and complexity to the clients.

Broke the interface contract - This is the cardinal rule for integration process management. The workgroup was purposefully setup to have a loosely coupled design process to validate if WebServices would work when the design teams had little interaction. For any integration process to be successful, there needs to be well-known interfaces and definitions setout to prescribe how systems will interoperate. In Web Services the WSDL is that definition. Amazingly, we actual achieved interoperability with different agencies following different patterns for the WSDL services interface; this capability presents a strong statement for the viability and flexibility of Web Services. Hey, It actually worked!!!!

## UDDI Publishing and Properties:

The workgroup's original objective with the UDDI was to have each agency publish their capabilities into a UDDI with properties describing what type of address changes that agency was ready to receive from other agencies. The process was supposed to work as follows:

4. Agency A published an Organization, definition and a Service definition for getAddress, ChangeAddress, and QueryAddress. Each Service would contain properties to identify if that service was subscribing to a certain type of address e.g. Mailing, Business, etc.
5. Agency B upon receiving an address change through its business processes would search the UDDI for a service that had the matching address type in its properties definitions.
6. When Agency B found Agency A's Service definition, Agency B would access Agency A's WSDL URI and dynamically invoke Agency A's service operation to change A's Address

While this loosely coupled process is the eventual promise of web services, we encountered several challenges that inhibited its total implementation. The issues are as follows:

- Visibility of properties attached to services
- Complexity of the UDDI programming model
- Complexity of dynamically invoking WSDL services
- Instability of UDDI

Visibility of properties attached to services - In a perfect UDDI world the Service publication would follow the WSDL definition and have one Service with multiple operations or "bindings" in UDDI terminology. The service would have a binding for the getAddress, ChangeAddress, and QueryAddress WSDL URI TModel<sup>3</sup> definitions. However, the browsing tools available do not navigate down into this level of property settings and TModel. If we had used a single UDDI Service such as "AddressChange" then the agencies would have had difficulty browsing down into the TModels to identify which Binding defined which address type and WSDL URI. As the project moved forward it became apparent that programmatic access to the low level TModels was not going to happen, this required having our UDDI definitions in a human readable structure.

Complexity of the UDDI programming model - The UDDI programming model is very flexible and has a very extensible structure. This flexibility however brings with it a high degree of complexity. The Object model has multiple locations to store "description", "URI", and "Properties". To say the least it is confusing, and best practices are not well defined. This complexity lead to several false starts and restarts in publishing to the UDDI. At first every agency had multiple organizations published, and then multiple services etc. In the end every agency did get their Organization, Services, and WSDLs published in the same structural manner, again it worked! However, most organizations used a manual browser to publish rather than programmatically publishing to the UDDI.

---

<sup>3</sup> A TModel is the UDDI definition of an Operation in the WSDL.

Complexity of dynamically invoking WSDL services - The workgroup's objective was to search the UDDI, retrieve the URI for a service's WSDL, dynamically consume that WSDL and dynamically bind to that WSDL. This proved much more complicated for all the vendors than originally thought. In my testing of the process, I could easily consume a WSDL and dynamically invoke the service operation if there were no complex data types. However, as stated earlier we had purposely introduced the AddressData complex data type to validate the ability to handle "real world" data. The AddressData type became a stumbling block to dynamic invocations using the JAX-RPC mechanism for standard Java Clients. In testing, I was able to dynamically invoke the AddressChange BEA WSDLs on the BEA WebLogic server web service implementation, but when attempting the same dynamic invocation process on other vendor services, it failed to work consistently. After a considerable effort to make the dynamic invocations work, I made the recommendation that the workgroup move back to the more reliable static client invocation. Note: The introduction of the Type "B" WSDL interfaces would have introduced more difficulty for dynamic invocations. With Type "A" WSDL interfaces, a service would do one search in the UDDI, get one WSDL and with that WSDL make the presumption that they could make multiple calls to getAddress, changeAddress, and queryAddress. With the type "B" WSDL interface, that presumption (valid by the original WSDL definition) would produce errors. The type "B" WSDL would also increase network traffic and processing time to consume multiple WSDL definitions.

Instability of UDDI - The Original UDDI installed at VipNet was not interoperable, and the current UDDI has not been stable. We are working with VipNet and the BEA OEM Company of the UDDI, Acumen, to find out why it has not been reliable. My suspicion is that they have not tested as extensively with programmatic clients from multiple vendors. The Workgroup has thrown several different vendor products at the UDDI; with shall we say an "ill-defined" access process, due to the complexity of the UDDI model. I suspect that one or several of us has introduced some data or a sequence of process calls that the Acumen UDDI had not anticipated, and can't handle... Our Log reports to Acumen should help them learn as well.

### 13. INTEROPERABILITY ISSUES

#### Status on each agency interoperability issues:

The following Table has several feature point matrixes with the Agencies success or compliance to the following feature points.

Feature	VABC	DMV	VRS	GMU	DHRM	DOE	Roanoke/ VA Beach
WSDL Interface "A"	Y	Y	N	N	Y	X	Y
WSDL Interface "B"	N	N	Y	Y	N	X	N

WSDL URL Access (Available)	Y	Y	Y	Y	Y	N	N <sup>4</sup>
BEA Client Software could consume the WSDL (in the case of Interface "B" each operation is listed)	Y	Y	N <sup>5</sup>	Y	Y		N
WSDL Port Access (Available)	Y	Y	N	Y	NT		
Published UDDI Organization with Single UDD Service	Y	Y	Y	Y	Y	Y	Y
Published Service Properties	Y					Y	
Published to UDDI Programmatically	Y	X	X	X	X	Y	X
Handled Complex AddressData (Service)	Y	Y		Y	NT		
Handled Complex Array Address Data (Service)	Y	Y		N <sup>6</sup>	NT		
Handled Complex Array Address Data (Client)	N <sup>7</sup>				NT		
Test Operations completed getAddress	Y	Y	NT	Y	NT		
Test Operations completed changeAddress	Y	Y	NT	Y	NT		
Test Operations completed queryAddress	Y	Y	N	N	NT		

Table 5 Agency Interoperability

Where:

- Y = Yes
- N = No
- X = Unknown
- NT = Not Tested, have not yet completed testing
- Blank = Indeterminate, a previous dependant feature failed, or I cannot validate.

#### 14. "BEST PRACTICE" COMMENTS

None at this time.

#### 15. OTHER CONCERNS / ISSUES

- UDDI must be implemented early in the process.
- Subscription services must be implemented early in the process.
- Address formats need to be identified.
- XML structures (list information) must be identified.
- Test data will need to be coordinated between teams.
- Unique identifier should be defined for companies.
- Unique identifier should be defined for individuals.
- SSN identifier may possibly be needed for a sole proprietor.

<sup>4</sup> Could access once, but not recently.

<sup>5</sup> Could consume changeAddress, and queryAddress, could not consume getAddress

<sup>6</sup> The BEA client returned a single AddressData object, not an array. The WSDL looks correct, we will investigate further.

<sup>7</sup> The BEA client returned an array with each element containing the last xml element returned. We are submitting a change request to BEA. There is another conversion mechanism that will correct this. I will investigate using it for the Test Harness when we get to COTS in September. "We are not afraid!!"

## 16. OTHER COMMENTS

In conclusion, WebService interoperability is a reality. The loosely coupled aspect of WebServices however is not mature. There are two aspects to the loosely coupled scenario. One is that design teams can work independent and rapidly consume web services from other design teams. This is not quite reality. The teams must have a well-known interface and definition of the services they want to share. They can definitely work in a looser coupling than CORBA, COM, and other legacy integration scenarios, but not to the level that we eventually need. The other loosely coupled aspect is the ability to late bind or dynamically bind to a WSDL at some previously unknown URI. This works if you use simple data types, but then you can only do simple interoperability and integration. There are efforts in the standard committees to move this forward but at this time it is not mature enough for the average working developer to successfully implement. The basis for this dynamic binding is the look up service provided by the UDDI. I believe that the current UDDI model is too complex, or at least the available tools are not powerful enough to make it accessible to the ordinary human. Web Services are viable for integration between closely coordinated organizations with agreed upon interfaces. They do provide a much easier and more interoperable integration technology than previously available. To roughly paraphrase Neal Armstrong: One large step forward for technology; a small step forward for full system integration.

## 17. COST/TIME ESTIMATE

**Meetings:** 6 hours

**Development by VABC:** 20 hours **Development by BEA:** ? hours

**Testing by VABC:** 10 hours **Testing by BEA:** ? hours

**Training:** 10 hours

## APPENDIX 1

```
<?xml version="1.0" encoding="utf-8" ?>
<definitions xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://whowery:7001" targetNamespace="http://whowery:7001"
xmlns="http://schemas.xmlsoap.org/wsdl/">
<types>
```



```
- <xsd:schema attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="java:language_builtins.util" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:stns="java:language_builtins.util">
- <xsd:complexType name="Vector">
- <xsd:complexContent>
- <xsd:restriction base="soapenc:Array"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
  <xsd:attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:anyType[]"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
  </xsd:restriction>
</xsd:complexContent>
</xsd:complexType>
</xsd:schema>
- <xsd:schema attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="java:com.bea.sesouth.webservices"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:stns
="java:com.bea.sesouth.webservices">
  <xsd:import namespace="java:language_builtins.util" />
- <xsd:complexType name="AddressLog">
- <xsd:sequence>
  <xsd:element name="addressVector" maxOccurs="1" type="tp:Vector" minOccurs="1"
nillable="true" xmlns:tp="java:language_builtins.util" />
  <xsd:element name="originator" maxOccurs="1" type="xsd:string" minOccurs="1"
nillable="true" />
</xsd:sequence>
</xsd:complexType>
- <xsd:complexType name="AddressData">
- <xsd:sequence>
  <xsd:element name="originator" maxOccurs="1" type="xsd:string" minOccurs="1"
nillable="true" />
  <xsd:element name="address2" maxOccurs="1" type="xsd:string" minOccurs="1"
nillable="true" />
  <xsd:element name="state" maxOccurs="1" type="xsd:string" minOccurs="1"
nillable="true" />
  <xsd:element name="name" maxOccurs="1" type="xsd:string" minOccurs="1"
nillable="true" />
  <xsd:element name="zip" maxOccurs="1" type="xsd:string" minOccurs="1"
nillable="true" />
  <xsd:element name="address1" maxOccurs="1" type="xsd:string" minOccurs="1"
nillable="true" />
  <xsd:element name="addressID" maxOccurs="1" type="xsd:string" minOccurs="1"
nillable="true" />
  <xsd:element name="city" maxOccurs="1" type="xsd:string" minOccurs="1"
nillable="true" />
  <xsd:element name="updateDate" maxOccurs="1" type="xsd:dateTime" minOccurs="1"
nillable="true" />
  <xsd:element name="addressType" maxOccurs="1" type="xsd:string" minOccurs="1"
nillable="true" />
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
</types>
- <message name="changeAddress">
  <part name="addressData" xmlns:partns="java:com.bea.sesouth.webservices"
type="partns:AddressData" />
  <part name="string" xmlns:partns="http://www.w3.org/2001/XMLSchema"
type="partns:string" />
```

```
</message>
- <message name="changeAddressResponse">
  <part name="result" xmlns:partns="http://www.w3.org/2001/XMLSchema"
type="partns:boolean" />
</message>
- <message name="getAddress">
  <part name="string" xmlns:partns="http://www.w3.org/2001/XMLSchema"
type="partns:string" />
  <part name="string0" xmlns:partns="http://www.w3.org/2001/XMLSchema"
type="partns:string" />
</message>
- <message name="getAddressResponse">
  <part name="result" xmlns:partns="java:com.bea.sesouth.webservices"
type="partns:AddressData" />
</message>
- <message name="queryAddress">
  <part name="date" xmlns:partns="http://www.w3.org/2001/XMLSchema"
type="partns:dateTime" />
  <part name="date0" xmlns:partns="http://www.w3.org/2001/XMLSchema"
type="partns:dateTime" />
  <part name="string" xmlns:partns="http://www.w3.org/2001/XMLSchema"
type="partns:string" />
  <part name="string0" xmlns:partns="http://www.w3.org/2001/XMLSchema"
type="partns:string" />
</message>
- <message name="queryAddressResponse">
  <part name="result" xmlns:partns="java:com.bea.sesouth.webservices"
type="partns:AddressLog" />
</message>
- <portType name="AddressChangeBeanPortType">
- <operation name="changeAddress">
  <input message="tns:changeAddress" />
  <output message="tns:changeAddressResponse" />
</operation>
- <operation name="getAddress">
  <input message="tns:getAddress" />
  <output message="tns:getAddressResponse" />
</operation>
- <operation name="queryAddress">
  <input message="tns:queryAddress" />
  <output message="tns:queryAddressResponse" />
</operation>
</portType>
- <binding name="AddressChangeBeanSoapBinding" type="tns:AddressChangeBeanPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
- <operation name="queryAddress">
  <soap:operation soapAction="" />
- <input>
  <soap:body use="encoded" namespace="http://whowery:7001"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
  </input>
- <output>
  <soap:body use="encoded" namespace="http://whowery:7001"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
  </output>
</operation>
- <operation name="changeAddress">
```

```
<soap:operation soapAction="" />
- <input>
  <soap:body use="encoded" namespace="http://whowery:7001"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
  </input>
- <output>
  <soap:body use="encoded" namespace="http://whowery:7001"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
  </output>
</operation>
- <operation name="getAddress">
  <soap:operation soapAction="" />
- <input>
  <soap:body use="encoded" namespace="http://whowery:7001"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
  </input>
- <output>
  <soap:body use="encoded" namespace="http://whowery:7001"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
  </output>
</operation>
</binding>
- <service name="AddressChangeBean">
- <documentation>todo: add your documentation here</documentation>
- <port name="AddressChangeBeanPort" binding="tns:AddressChangeBeanSoapBinding">
  <soap:address
location="http://localhost:7001/state.va.ws.addresschange/AddressChangeBean?WSDL" />
  </port>
</service>
</definitions>
```

## Test Plan

## Revision History

Date	Version	Author	Description
07-01-02	1.0		Initial Draft

## Introduction

### Purpose and Scope

This document specifies the objectives and approach for conducting test for the Web Services Workgroup Proof-Of-Concept. A high-level test strategy and high level test case scenarios are outlined in this document.

### List of Reference Documents

	Document Name / Location	Document Description
1	VABC_BEA WS POC.doc	This proof-of-concept document outlines team

		member names,/roles, a high level design of the application, hardware and software requirements, and miscellaneous project comments.
--	--	--

## Testing Approach and Objectives

### Scope of Proof-Of-Concept Tests

The tests will focus on the functionality and implementation of key web services features. The limited business logic that utilizes the web service features will also be tested but with less emphasis. The tests will review data flows between the application and the web services; as well as, monitoring for any potential problems in performance or bottlenecks. Several test cycles are anticipated and testing will continue until satisfactory results are achieved.

## Individual Steps and Test Descriptions

### System Wide Functional Test Descriptions

#### Tested Features

Scenario / Test Case Name	Test Case Document(s)
Test get Address functionality via services	
Test methods associated with changing or updating an address	
Test queryAddress functionality via services.	
Test ability to register services in the UDDI.	
Test client look up of UDDI Services.	
Test invokes a service on the WSDL from the UDDI.	
Test performance on a queryAddress functionality using the services.	
Test Another Agency for changing an address	
Validation test of Agency Logs.	

#### Test gets Address functionality via services.

The JUnit test will call the getAddress WebService to retrieve a known address from the VABC test database.

**Prerequisites:** Have a known address in the test database.

- Statically bind to the WSDL for the getAddress WebService
- Invoke the getAddress WebService with the known AddressID
- Retrieve the getAddress data and write it to a log for validation.

#### Test methods associated with changing or updating an address

The JUnit test will call the getAddress WebService to retrieve a known address from the VABC test database.

**Prerequisites:** getAddress must function; Have a known address in the test database.

- Statically bind to the WSDL for the changeAddress WebService
- Invoke the changeAddress WebService with the known AddressID, but with different street and city info
- Retrieve the same address with getAddress data and write it to a log for validation. The data should have changed
- Invoke the changeAddress WebService with a new address.

- Retrieve the same address with getAddress data and write it to a log for validation. The data should be there.

#### **Test queryAddress functionality via services.**

The Unit test will call the queryAddress service with a data range to receive a known set of address from the log.

**Prerequisites:** A known set of data is in the test database to validate the correct return addresses.

- Statically bind to the WSDL for the queryAddress WebService
- Invoke the queryAddress service with a defined date range and address type.
- Retrieve the set off addresses
- Write the address to a log for validation
- Repeat the process for another address type and alternate date ranges.
- Run the test for changeAddress to insert a new address
- Repeat the test for the current date and address type as used in changeAddress
- Validate that the addresses updated and added in changeAddress are in the log query results.
- Write the results to a log file for validation.

#### **Test ability to register services in the UDDI.**

Client software must be able to publish to the UDDI register and set properties in the Category Bag to identify the agency name, service, type of addresses to receive, and the WSDL URL for the service to provide the address get, change and query operations.

A JUnit test will be written that will do the following

- Publish a set of address services to the UDDI registry,
- Search for these same services.
- Write to a log file for validation that the services were in fact registered.
- Delete the test services from the UDDI registry

#### **Test client look up of UDDI Services.**

The client software must be able to search the UDDI registry for a given address type, retrieve the WSDL URL. This test will be executed and validated as the search portion of the register services to UDDI test.

#### **Test invokes a service on the WSDL from the UDDI.**

The service must be able to retrieve a WSDL from the UDDI search and with this WSDL, Bind to the web service, invoke the WebService and return the results of the WebService.

This test will require that the getAddress functionality and the UDDI Search functionality is already functioning. The test will do the following:

**Prerequisites:** UDDI Search works, getAddress WS works.

- Publish a UDDI entry for the getAddress service. Set one of the categoryBag properties to "VABCTest".
- Search for the VABCTest getAddress service in the UDDI
- Retrieve the VABCTest getAddress WSDL
- Bind to the WSDL end point
- Invoke the getAddress service and retrieve a known address in the VABC site.
- Write the retrieved address to a log for validation that the test completed correctly.
- Delete the VABCTest getAddress service from the UDDI

#### **Test performance on a queryAddress functionality using the services.**

Write a JUnit test to retrieve addresses from QueryAddress with increasing date ranges. The objective of the test will be to determine a performance curve for the number of addresses returned.

#### **Test Another Agency for changing an address.**

Create a JUnit test to test that address changes propagated to another agency did in fact occur.

**Caution:** This test can only be executed in a controlled test environment. Potential side effects in target agency data.

**Prerequisite:** Another agency has subscribed for a business address.

- Create a dummy Address for Testing purposes  
Name=VABCDUMMY Address  
AddressID=99955999

Street1=street (N)  
City=Richmond

- Search the UDDI registry for an agency accepting business address
- Invoke a getAddress on the target agency for the AddressID of our dummy address.
- If the address exists, validated that the name is "VABCDUMMY Address" if not decrement the AddressID and try again.
- If not found invoke changeAddress to insert the address.
- If found increment the Street (N) number and invoke changeAddress to update the address
- Invoke getAddress to retrieve the "VABCDUMMY Address"
- Validate that the Street1 equals the update or inserted value. Write results to a log
- Write the results to a Log file for Validation
- Repeat the test on the next agency subscribing to business addresses.

### Validation test of Agency Logs.

Validate Log entries between agencies.

- VABC gets an address change for a Business address.
- VABC looks into the UDDI to get the list of agencies that have subscribed for Business Addresses e.g. VRS, VMS.
- VABC makes an entry into their log of the address data sent, with the Source being VABC.
- VABC sends the address via the changeAddress method to VRS and VMS
- VRS and VMS receive the address and record the entry in their respective logs. With the source being VABC
- During Validation the VABC log is interrogated and the entry is found with a source VABC (source=agency Log owner means a sent address). Then the agencies registered for Business addresses logs are checked to validate that they received the given address with the VABC sender.

This process will validate that addresses sent were actually received by the subscribing agencies.

### Problem Recording and Resolution

The VABC/BEA Team will compile problems and Issues that either suspend testing or require a restart of testing into a separate document for review.

### Rework, Review, and Retest Procedures

Upon completion of testing and upon update of all test documentation, a review will be held between the VABC/BEA Team to discuss test results any rework, review, retest, restart or suspensions. This information will be documented, along with supporting test results and will be made available to the workgroup.

### Suspension, Restart, and Exit Criteria

System tests between will be suspended and cannot continue if the following conditions occur:

- N/A

### Success Criteria

Tests will run until such time that all test case scenarios are completed successfully.

## Appendix H: Team 3 Final Report – VRS, DHRM, SilverStream, & Mitem

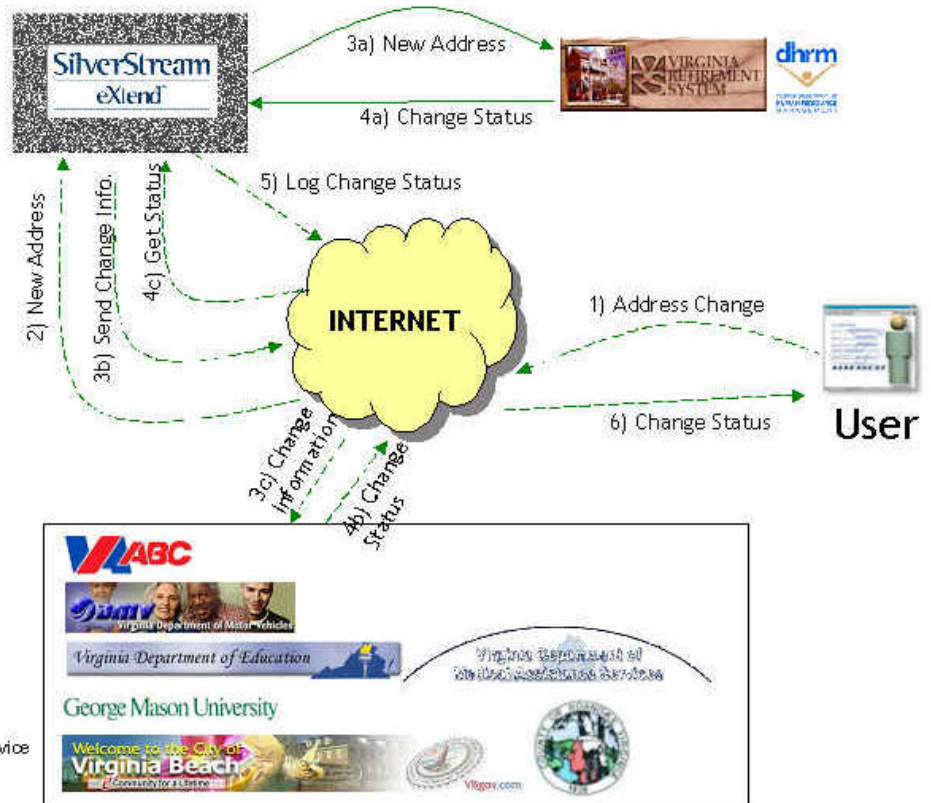
<div>1.</div> <div>PROJECT PLAN (SCHEDULE)</div>	<div>Attach high-level project plan with list of meetings, time and location giving percent complete.</div> <div><table><tr><th rowspan="2">ID</th><th rowspan="2">Task Name</th><th rowspan="2">Duration</th><th rowspan="2">Start</th><th rowspan="2">Finish</th><th colspan="4">June</th><th colspan="4">July</th><th colspan="4">August</th></tr><tr><th>5/26</th><th>6/2</th><th>6/9</th><th>6/16</th><th>6/23</th><th>6/30</th><th>7/7</th><th>7/14</th><th>7/21</th><th>7/28</th><th>8/4</th><th>8/11</th></tr><tr><td>1</td><td>COTS Meetings</td><td>40 days</td><td>Thu 5/30/02</td><td>Mon 8/5/02</td><td colspan="12"><div></div></td></tr><tr><td>7</td><td>Workgroup Report</td><td>56 days</td><td>Fri 5/31/02</td><td>Fri 8/16/02</td><td colspan="12"><div></div></td></tr><tr><td>8</td><td>First Draft</td><td>12 days</td><td>Fri 5/31/02</td><td>Mon 6/17/02</td><td colspan="12"><div></div></td></tr><tr><td>18</td><td>Second Draft</td><td>15 days</td><td>Tue 6/25/02</td><td>Mon 7/15/02</td><td colspan="12"><div>BC,ZO,CH,MR</div></td></tr><tr><td>19</td><td>Final</td><td>10 days</td><td>Fri 7/19/02</td><td>Fri 8/16/02</td><td colspan="12"><div></div></td></tr><tr><td>21</td><td>Prototype Development</td><td>40 days</td><td>Mon 6/3/02</td><td>Fri 7/26/02</td><td colspan="12"><div></div></td></tr><tr><td>22</td><td>VRS Web Services</td><td>36 days</td><td>Mon 6/3/02</td><td>Mon 7/22/02</td><td colspan="12"><div>BC,ZO,MR</div></td></tr><tr><td>23</td><td>HWS/V Reqs Meeting</td><td>4 hrs</td><td>Mon 6/3/02</td><td>Mon 6/3/02</td><td colspan="12"><div>BC,ZO,MR</div></td></tr><tr><td>24</td><td>Installation</td><td>8 hrs</td><td>Mon 6/10/02</td><td>Mon 6/10/02</td><td colspan="12"><div>MR</div></td></tr><tr><td>25</td><td>Bus Fn Reqs Meetings</td><td>4 hrs</td><td>Mon 6/10/02</td><td>Mon 6/10/02</td><td colspan="12"><div>MR</div></td></tr><tr><td>26</td><td>Build Login Component</td><td>8 hrs</td><td>Tue 6/11/02</td><td>Tue 6/11/02</td><td colspan="12"><div>MR</div></td></tr><tr><td>27</td><td>Build VRS Web Services</td><td>20.5 days</td><td>Wed 6/12/02</td><td>Wed 7/10/02</td><td colspan="12"><div></div></td></tr><tr><td>34</td><td>Unit Testing of VRS Web Services</td><td>80 hrs</td><td>Tue 7/9/02</td><td>Mon 7/22/02</td><td colspan="12"><div>MR</div></td></tr><tr><td>35</td><td>DHRM Web Services</td><td>40 days</td><td>Mon 6/3/02</td><td>Fri 7/26/02</td><td colspan="12"><div>BC,ZO,MR</div></td></tr><tr><td>36</td><td>HWS/V Reqs Meeting</td><td>4 hrs</td><td>Mon 6/3/02</td><td>Mon 6/3/02</td><td colspan="12"><div>BC,ZO,MR</div></td></tr><tr><td>37</td><td>Installation</td><td>4 hrs</td><td>Fri 6/7/02</td><td>Fri 6/7/02</td><td colspan="12"><div>MR</div></td></tr><tr><td>38</td><td>Bus Fn Reqs Meetings</td><td>4 hrs</td><td>Fri 6/7/02</td><td>Fri 6/7/02</td><td colspan="12"><div>MR</div></td></tr><tr><td>39</td><td>Build Login Component</td><td>8 hrs</td><td>Fri 6/7/02</td><td>Mon 6/10/02</td><td colspan="12"><div></div></td></tr><tr><td>40</td><td>Build DHRM Web Services</td><td>23.5 days</td><td>Thu 6/13/02</td><td>Tue 7/16/02</td><td colspan="12"><div></div></td></tr><tr><td>47</td><td>Unit Testing of DHRM Web Services</td><td>80 hrs</td><td>Mon 7/15/02</td><td>Fri 7/26/02</td><td colspan="12"><div>MR</div></td></tr><tr><td>48</td><td>Integration Testing</td><td>120 hrs</td><td>Mon 7/29/02</td><td>Fri 8/16/02</td><td colspan="12"><div></div></td></tr><tr><td>49</td><td>Training</td><td>80 hrs</td><td>Mon 8/5/02</td><td>Fri 8/16/02</td><td colspan="12"><div></div></td></tr></table></div>	ID	Task Name	Duration	Start	Finish	June				July				August				5/26	6/2	6/9	6/16	6/23	6/30	7/7	7/14	7/21	7/28	8/4	8/11	1	COTS Meetings	40 days	Thu 5/30/02	Mon 8/5/02	<div></div>												7	Workgroup Report	56 days	Fri 5/31/02	Fri 8/16/02	<div></div>												8	First Draft	12 days	Fri 5/31/02	Mon 6/17/02	<div></div>												18	Second Draft	15 days	Tue 6/25/02	Mon 7/15/02	<div>BC,ZO,CH,MR</div>												19	Final	10 days	Fri 7/19/02	Fri 8/16/02	<div></div>												21	Prototype Development	40 days	Mon 6/3/02	Fri 7/26/02	<div></div>												22	VRS Web Services	36 days	Mon 6/3/02	Mon 7/22/02	<div>BC,ZO,MR</div>												23	HWS/V Reqs Meeting	4 hrs	Mon 6/3/02	Mon 6/3/02	<div>BC,ZO,MR</div>												24	Installation	8 hrs	Mon 6/10/02	Mon 6/10/02	<div>MR</div>												25	Bus Fn Reqs Meetings	4 hrs	Mon 6/10/02	Mon 6/10/02	<div>MR</div>												26	Build Login Component	8 hrs	Tue 6/11/02	Tue 6/11/02	<div>MR</div>												27	Build VRS Web Services	20.5 days	Wed 6/12/02	Wed 7/10/02	<div></div>												34	Unit Testing of VRS Web Services	80 hrs	Tue 7/9/02	Mon 7/22/02	<div>MR</div>												35	DHRM Web Services	40 days	Mon 6/3/02	Fri 7/26/02	<div>BC,ZO,MR</div>												36	HWS/V Reqs Meeting	4 hrs	Mon 6/3/02	Mon 6/3/02	<div>BC,ZO,MR</div>												37	Installation	4 hrs	Fri 6/7/02	Fri 6/7/02	<div>MR</div>												38	Bus Fn Reqs Meetings	4 hrs	Fri 6/7/02	Fri 6/7/02	<div>MR</div>												39	Build Login Component	8 hrs	Fri 6/7/02	Mon 6/10/02	<div></div>												40	Build DHRM Web Services	23.5 days	Thu 6/13/02	Tue 7/16/02	<div></div>												47	Unit Testing of DHRM Web Services	80 hrs	Mon 7/15/02	Fri 7/26/02	<div>MR</div>												48	Integration Testing	120 hrs	Mon 7/29/02	Fri 8/16/02	<div></div>												49	Training	80 hrs	Mon 8/5/02	Fri 8/16/02	<div></div>											
ID	Task Name						Duration	Start	Finish	June				July				August																																																																																																																																																																																																																																																																																																																																																																																																		
		5/26	6/2	6/9	6/16	6/23				6/30	7/7	7/14	7/21	7/28	8/4	8/11																																																																																																																																																																																																																																																																																																																																																																																																				
1	COTS Meetings	40 days	Thu 5/30/02	Mon 8/5/02	<div></div>																																																																																																																																																																																																																																																																																																																																																																																																															
7	Workgroup Report	56 days	Fri 5/31/02	Fri 8/16/02	<div></div>																																																																																																																																																																																																																																																																																																																																																																																																															
8	First Draft	12 days	Fri 5/31/02	Mon 6/17/02	<div></div>																																																																																																																																																																																																																																																																																																																																																																																																															
18	Second Draft	15 days	Tue 6/25/02	Mon 7/15/02	<div>BC,ZO,CH,MR</div>																																																																																																																																																																																																																																																																																																																																																																																																															
19	Final	10 days	Fri 7/19/02	Fri 8/16/02	<div></div>																																																																																																																																																																																																																																																																																																																																																																																																															
21	Prototype Development	40 days	Mon 6/3/02	Fri 7/26/02	<div></div>																																																																																																																																																																																																																																																																																																																																																																																																															
22	VRS Web Services	36 days	Mon 6/3/02	Mon 7/22/02	<div>BC,ZO,MR</div>																																																																																																																																																																																																																																																																																																																																																																																																															
23	HWS/V Reqs Meeting	4 hrs	Mon 6/3/02	Mon 6/3/02	<div>BC,ZO,MR</div>																																																																																																																																																																																																																																																																																																																																																																																																															
24	Installation	8 hrs	Mon 6/10/02	Mon 6/10/02	<div>MR</div>																																																																																																																																																																																																																																																																																																																																																																																																															
25	Bus Fn Reqs Meetings	4 hrs	Mon 6/10/02	Mon 6/10/02	<div>MR</div>																																																																																																																																																																																																																																																																																																																																																																																																															
26	Build Login Component	8 hrs	Tue 6/11/02	Tue 6/11/02	<div>MR</div>																																																																																																																																																																																																																																																																																																																																																																																																															
27	Build VRS Web Services	20.5 days	Wed 6/12/02	Wed 7/10/02	<div></div>																																																																																																																																																																																																																																																																																																																																																																																																															
34	Unit Testing of VRS Web Services	80 hrs	Tue 7/9/02	Mon 7/22/02	<div>MR</div>																																																																																																																																																																																																																																																																																																																																																																																																															
35	DHRM Web Services	40 days	Mon 6/3/02	Fri 7/26/02	<div>BC,ZO,MR</div>																																																																																																																																																																																																																																																																																																																																																																																																															
36	HWS/V Reqs Meeting	4 hrs	Mon 6/3/02	Mon 6/3/02	<div>BC,ZO,MR</div>																																																																																																																																																																																																																																																																																																																																																																																																															
37	Installation	4 hrs	Fri 6/7/02	Fri 6/7/02	<div>MR</div>																																																																																																																																																																																																																																																																																																																																																																																																															
38	Bus Fn Reqs Meetings	4 hrs	Fri 6/7/02	Fri 6/7/02	<div>MR</div>																																																																																																																																																																																																																																																																																																																																																																																																															
39	Build Login Component	8 hrs	Fri 6/7/02	Mon 6/10/02	<div></div>																																																																																																																																																																																																																																																																																																																																																																																																															
40	Build DHRM Web Services	23.5 days	Thu 6/13/02	Tue 7/16/02	<div></div>																																																																																																																																																																																																																																																																																																																																																																																																															
47	Unit Testing of DHRM Web Services	80 hrs	Mon 7/15/02	Fri 7/26/02	<div>MR</div>																																																																																																																																																																																																																																																																																																																																																																																																															
48	Integration Testing	120 hrs	Mon 7/29/02	Fri 8/16/02	<div></div>																																																																																																																																																																																																																																																																																																																																																																																																															
49	Training	80 hrs	Mon 8/5/02	Fri 8/16/02	<div></div>																																																																																																																																																																																																																																																																																																																																																																																																															
<div>2.</div> <div>LIST OF TEAM NAME, MEMBERS, ROLES AND RESPONSIBILITIES</div>	<div>Statement of team member roles and responsibilities – who is doing what. Highlight new/additional team members being brought in to assist since last report.</div> <div><table><tr><td>VRS</td></tr><tr><td>Sonja Korb – point of contact/project lead</td></tr><tr><td>John Oliver – network engineer</td></tr><tr><td>DHRM</td></tr><tr><td>Belchoir Mira – point of contact/project lead</td></tr><tr><td>Bradley Paul – computer systems senior engineer</td></tr><tr><td>Allen Kass – network administrator</td></tr><tr><td>SilverStream</td></tr><tr><td>Chris Holcombe – point of contact</td></tr><tr><td>Barbara Cain – service manager</td></tr><tr><td>Tanzeena O'Brien – technical manager</td></tr><tr><td>Mark Romiti – engineer</td></tr><tr><td>Mitem</td></tr><tr><td>Dave Pendergrass – regional sales manager</td></tr></table></div>	VRS	Sonja Korb – point of contact/project lead	John Oliver – network engineer	DHRM	Belchoir Mira – point of contact/project lead	Bradley Paul – computer systems senior engineer	Allen Kass – network administrator	SilverStream	Chris Holcombe – point of contact	Barbara Cain – service manager	Tanzeena O'Brien – technical manager	Mark Romiti – engineer	Mitem	Dave Pendergrass – regional sales manager																																																																																																																																																																																																																																																																																																																																																																																																					
VRS																																																																																																																																																																																																																																																																																																																																																																																																																				
Sonja Korb – point of contact/project lead																																																																																																																																																																																																																																																																																																																																																																																																																				
John Oliver – network engineer																																																																																																																																																																																																																																																																																																																																																																																																																				
DHRM																																																																																																																																																																																																																																																																																																																																																																																																																				
Belchoir Mira – point of contact/project lead																																																																																																																																																																																																																																																																																																																																																																																																																				
Bradley Paul – computer systems senior engineer																																																																																																																																																																																																																																																																																																																																																																																																																				
Allen Kass – network administrator																																																																																																																																																																																																																																																																																																																																																																																																																				
SilverStream																																																																																																																																																																																																																																																																																																																																																																																																																				
Chris Holcombe – point of contact																																																																																																																																																																																																																																																																																																																																																																																																																				
Barbara Cain – service manager																																																																																																																																																																																																																																																																																																																																																																																																																				
Tanzeena O'Brien – technical manager																																																																																																																																																																																																																																																																																																																																																																																																																				
Mark Romiti – engineer																																																																																																																																																																																																																																																																																																																																																																																																																				
Mitem																																																																																																																																																																																																																																																																																																																																																																																																																				
Dave Pendergrass – regional sales manager																																																																																																																																																																																																																																																																																																																																																																																																																				
<div>3.</div> <div>PROOF-OF- CONCEPT</div>	<div>Attach general design of your application narrative, flow diagrams, etc. – whatever you think will adequately convey the team’s approach in terms of design.</div>																																																																																																																																																																																																																																																																																																																																																																																																																			

DESIGN

**VRS/DHRM Users – Change Address:**

Change Address Process

- User initiates an address change via Web Service
- SilverStream eXtend receives that request and does two things in parallel
  - Sends the new address to VRS/DHRM
- Sends the change information to the other agencies via Web Service
  - Once the VRS/DHRM and other agencies return status messages,
  - Then the application will return all the statuses from all the agencies of the address change



**Figure 1 - VRS/DHRM Users – Change Address**

**Other Agency Users – Change Address:**

Change Address Process

- Another agency will initiate an address change via Web Service (machine to machine Web Service)
- VRS/DHRM will authenticate the individual making the change by making sure the user is in the Security DB.
  - If in the Security DB then
    - SilverStream eXtend receives that request and sends the new address to VRS/DHRM
    - The VRS/DHRM system returns status messages,



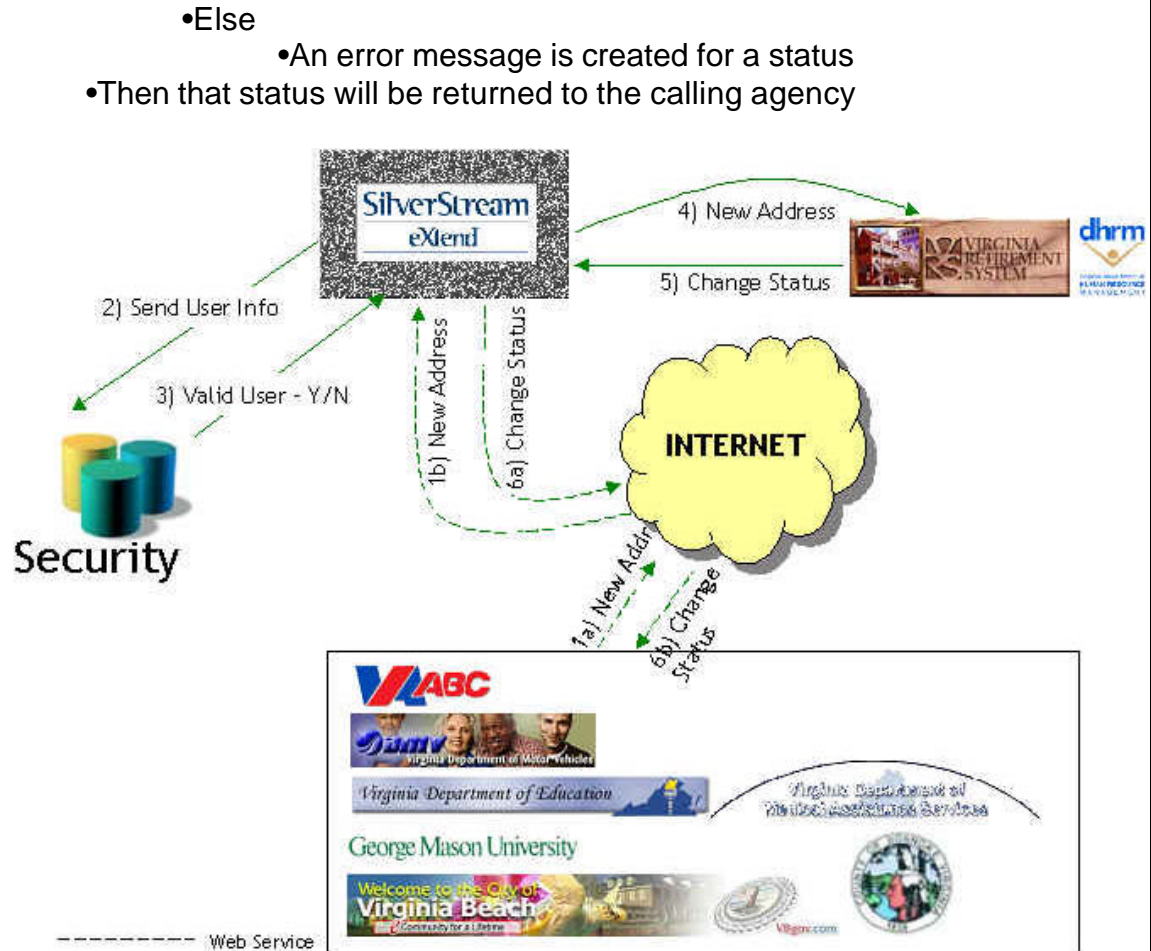


Figure 2 - Other Agency Users – Change Address

### Get Current Address by UserID:

#### Get Current Address by UserID

- Another agency will initiate a request to get the current address of a user via Web Service (machine to machine Web Service)
- VRS/DHRM will authenticate the addressed individual by making sure the user is in the Security DB.
  - If in the Security DB then
- SilverStream eXtend will extract that information from the back end system
- Else
  - An error message is created for a status
- Then the application will return the current address/status of a specified user to the requesting agency.



- Another agency will initiate a request to get all the changed addresses between a start date and an end date (machine to machine Web Service).
- VRS/DHRM will query the log file for each address change that has been propagated between the dates.
- For each address change, SilverStream eXtend will extract that information from the back end system.
- Then the application will return all the changed addresses between the dates to the requesting agency.

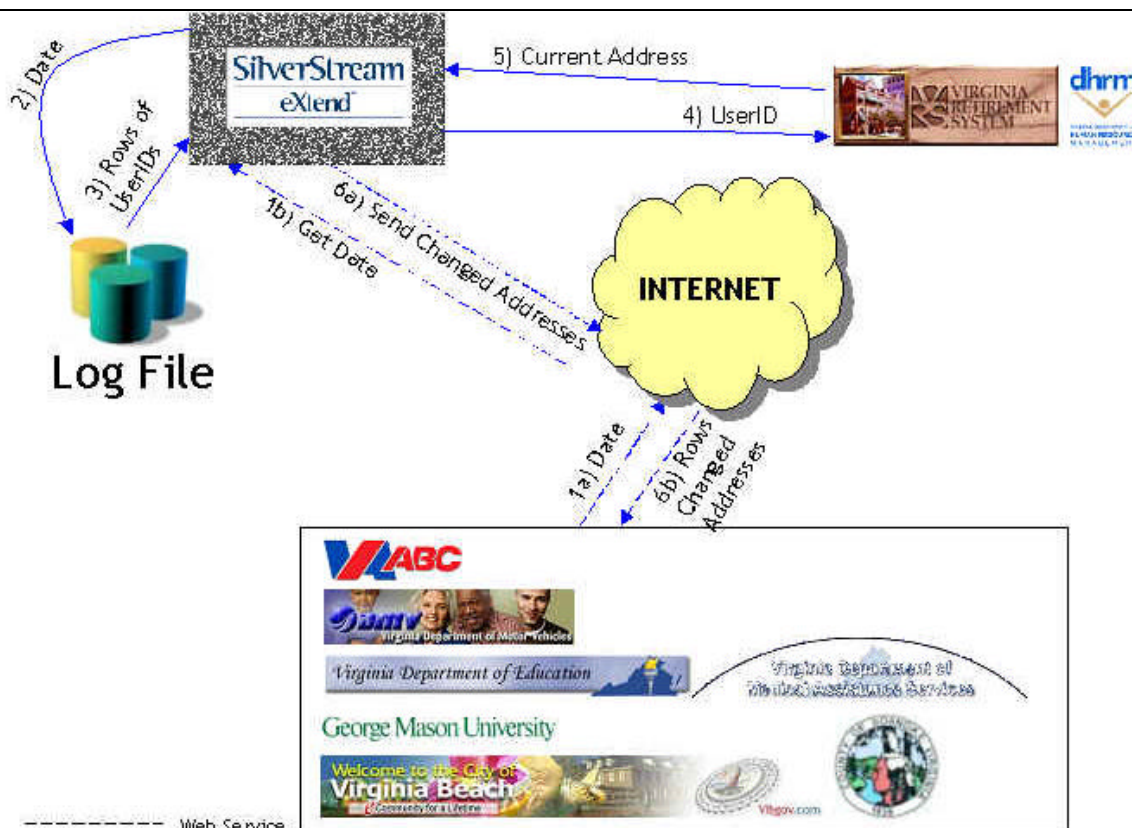


Figure 4 - Get Current Address by Date

### AddressLog Table Description:

Column Name	Data Type
PrimaryID	Numeric
	VarChar
<b>ACTION</b>	
OriginatorAgency	VarChar
AddressType	VarChar
AddressID	Numeric
TimeOfChange	DateTime
ErrorStatus	VarChar
Error	Boolean

### COLUMN NAME DESCRIPTION

PrimaryID This will be the primary key

Action The WS request was to GET or UPDATE address

OriginatorAgency The will the agency name where the address change originated

AddressType Home or Mailing address

AddressID UserID of the person changing the name

TimeOfChange When the address was changed

ErrorStatus Status of update or an error message

	<p>Error      Was there an error – True or False</p> <p>XXVI.    Get Log Information from AddressLog Table GetLogInfo(Date timestamp) returns a Collection of AddressLog Object</p> <p>AddressLog Class: OriginatorAgency (String), AddressType (String), AddressID (Numeric), TimeOfChange (DateTime) ErrorStatus (String)</p> <p>Query: SELECT .. FROM ADDRESSLOG WHERE TimeOfChange &gt; :argument</p> <p>XXVII.   Insert Log Information into AddressLog Table</p> <p><b>INSERTLOGINFO(ADDRESS ADDRESSLOG) SEE ABOVE CLASS DESCRIPTION</b> returns String ErrorStatus</p> <p>query: INSERT INTO ADDRESSLOG VALUES (..)</p> <p><b>COTSUser Table Description:</b> DHRM: This table will map an incoming SSN/ADDRESSID to an SSN/ADDRESSID that the mainframe can "understand". This was determined to be needed after DHRM looked at the test data.</p> <table border="1" data-bbox="410 1270 1563 1383"> <thead> <tr> <th>Column Name</th><th>Data Type</th></tr> </thead> <tbody> <tr> <td>AddressID</td><td>Numeric</td></tr> <tr> <td>UserID</td><td>VarChar</td></tr> </tbody> </table> <p><b>Localities Table Description:</b> DHRM: This table will support the city to locality mapping:</p> <table border="1" data-bbox="410 1501 1563 1663"> <thead> <tr> <th>Column Name</th><th>Data Type</th></tr> </thead> <tbody> <tr> <td><b>LOCALITY CODES</b></td><td>VarChar</td></tr> <tr> <td>Locality Name</td><td>VarChar</td></tr> </tbody> </table>	Column Name	Data Type	AddressID	Numeric	UserID	VarChar	Column Name	Data Type	<b>LOCALITY CODES</b>	VarChar	Locality Name	VarChar
Column Name	Data Type												
AddressID	Numeric												
UserID	VarChar												
Column Name	Data Type												
<b>LOCALITY CODES</b>	VarChar												
Locality Name	VarChar												
<p><b>4. BUSINESS/FUNCTIONAL REQUIREMENT</b></p>	<p>Statement of the specific business/functional requirements that are driving the team's application design.</p> <p>Implement the following Web Services for Virginia Retirement System and Department</p>												

S	<p>of Human Resources Management:</p> <ul style="list-style-type: none"> <li>• Change Address – update address information of a user.</li> <li>• Get Address – retrieve address information of a user.</li> <li>• Get Batch Address – retrieve rows of addresses that were changed between a start and end date.</li> </ul> <p>Change and get address enablement of the VRS system through a 3270 interface. Change and get address enablement of the DHRM system through the Mitem interface.</p> <p>Distribute a Change of Address request to the other agencies participating this Proof-Of-Concept.</p>
5. PROJECT EQUIPMENT DESCRIPTION	<p>Full description of the hardware and software to be used, who is providing it and where it is located.</p> <p>Hardware</p> <ul style="list-style-type: none"> <li>• 1 Windows NT Workstation/Server 4.0 or Windows 2000 provided by VRS</li> <li>• 1 Windows NT Workstation/Server 4.0 or Windows 2000 provided by DHRM <ul style="list-style-type: none"> <li>◦ Both machines must have, at a minimum, 512 MB RAM and 260 MB disk space</li> </ul> </li> <li>• Both machines will be given external access so that other agencies can access these machines.</li> </ul> <p>Software – VRS and DHRM will each get the following, provided by SilverStream</p> <ul style="list-style-type: none"> <li>• 1 Not-for-resale SilverStream eXtend Application Server 3.7.4</li> <li>• 1 Not-for-resale SilverStream eXtend Composer Enterprise 3.5</li> <li>• 1 Not-for-resale SilverStream eXtend Composer Developer 3.5</li> <li>• 1 Not-fore-resale SilverStream eXtend Composer 3270 Connector (for VRS)</li> </ul>
6. ACQUISITION AND INSTALLATION ACTIVITIES	<p>Indicate whether additional hardware and software were purchased or acquired for the proof-of-concept.</p> <p>VRS/DHRM</p> <p>To support marshalling of complex types in a Web Service, both agencies were upgraded to the latest and greatest version of SilverStream products: SilverStream 3.7.5, Composer 4.0, Workbench 4.0, and Jbroker Web 2.0 have been installed on server machine.</p>
7. SERVER REQUIREMENTS	<p>Describe changes to existing server or acquisition that will be needed before testing can begin. Describe the factors that influenced your decisions.</p> <p>NONE</p>
8. NETWORKING REQUIREMENTS	<p>Describe changes to existing network or procurement that will be needed before testing can begin. Describe the factors that influenced your decisions.</p> <p>NONE</p>
9. TRAINING REQUIREMENTS	<p>Identify the technical and business training that has been or will be needed to prepare staff for proof-of-concept project.</p> <p>SilverStream will be dependent on both agencies personnel for infrastructure support, and Mitem support at DHRM.</p>

	<p>Training on SilverStream products:</p> <ul style="list-style-type: none"><li>○ Application server</li><li>○ Composer Server</li><li>○ Composer Designer</li><li>○ 3270 Connector (VRS)</li><li>○ Workbench IDE</li></ul> <p>Training on the following concepts:</p> <ul style="list-style-type: none"><li>○ Java, HTML, SOAP, WSDL, Web Services, XML, and XSL</li></ul> <p>Mark Romiti showed Bradley the Workbench tool including how to consume a WSDL, the entity bean wizard, and how to create a service. He also gave him a final demo of the software showing the interoperability of DHRM and VRS.</p> <p>Mark is currently training Sonja more formally. She has already had a demo of the software showing interoperability between VRS, DHRM, DMV, and GMU.</p>																				
10. TESTING PLAN	<p>Attach a list showing all functions/processes to be tested. This should include descriptions of all testing scenarios and results.</p> <table><tr><th>Action</th><th>Result</th></tr><tr><td></td><td></td></tr><tr><td><b>Login with an invalid UserID</b></td><td></td></tr><tr><td>Go to POC Web Site</td><td>See the POC login screen</td></tr><tr><td>Login with an invalid UserID</td><td>See a page stating that the UserID/Password is incorrect</td></tr><tr><td></td><td></td></tr><tr><td><b>Login with an invalid Password</b></td><td></td></tr><tr><td>Go to POC Web Site</td><td>See the POC login screen</td></tr><tr><td>Login with an invalid Password</td><td>See a page stating that the UserID/Password is incorrect</td></tr><tr><td></td><td></td></tr></table>	Action	Result			<b>Login with an invalid UserID</b>		Go to POC Web Site	See the POC login screen	Login with an invalid UserID	See a page stating that the UserID/Password is incorrect			<b>Login with an invalid Password</b>		Go to POC Web Site	See the POC login screen	Login with an invalid Password	See a page stating that the UserID/Password is incorrect		
Action	Result																				
<b>Login with an invalid UserID</b>																					
Go to POC Web Site	See the POC login screen																				
Login with an invalid UserID	See a page stating that the UserID/Password is incorrect																				
<b>Login with an invalid Password</b>																					
Go to POC Web Site	See the POC login screen																				
Login with an invalid Password	See a page stating that the UserID/Password is incorrect																				
Web Services Workgroup	<table><tr><td>Final Report</td><td>Page 134</td><td>September 13, 2002</td></tr><tr><td><b>Change address from POC Web Site</b></td><td></td><td></td></tr></table>	Final Report	Page 134	September 13, 2002	<b>Change address from POC Web Site</b>																
Final Report	Page 134	September 13, 2002																			
<b>Change address from POC Web Site</b>																					

		Insert a log record containing information about the originating agency, address type, address ID, datetime of change, and status/errors.
		Receive a status back from the other agencies.
		Collate all the status messages and send back to user.
		See the status page containing the statuses of the address change from the current agency and the propagated agencies.
	<b>Change Address Web Service request from another agency of an individual who does NOT exist within the system.</b>	
	Receive a Web Service request from another agency of an address change of an individual who does NOT exist within the system.	Check the security DB to see if the individual exists within the system.
		Send an error message back to the agency requesting the address change stating the user does not exist within the system.
	<b>Change Address Web Service request from another agency of an individual who does exist within the system.</b>	
	Receive a Web Service request from another agency of an address change of an individual who does exist within the system.	Check the security DB to see if the individual exists within the system.
		Send the address change to the back-end system.
		Send the status back to the agency requesting the address change.
		Insert a log record containing information about the originating agency, address type, address ID, datetime of change, and status/errors.
	<b>Get Address Web Service request from another agency of an individual who is NOT in our system.</b>	
	Receive a Web Service request from another agency of an address of an individual who is NOT in our system.	Query the Security DB with the UserID.
		Send an error message back to the agency requesting the address stating the user does not exist within the system.
	<b>Get Address Web Service request from another agency of an individual who is in our system.</b>	
	Receive a Web Service request from another agency of an address of an individual who is in our system.	Query the Security DB with the UserID.
		Query the back-end system with the UserID.
		Send the address of that individual back to the agency requesting the address change.
	<b>Get Addresses Web Service request from another agency after a certain datetime.</b>	

	Receive a Web Service request from another agency for a list of address after a certain datetime.	Query the Log table of all entries where the datetime is after the requested date and propagated is true.
		For each record within the log file, get the AddressID and Query the back-end system with the AddressID (UserID).
		Send the rows of changed addresses after a certain datetime back to the requesting agency.
11. DEVELOPMENT/TECHNICAL	High-level description of specific development/technical issues and/or difficulties encountered during coding.	



						on ZIP code entered.	
	7/1/2002	DHRM: need to map an incoming SSN/ADDRESSID to an SSN/ADDRESSID on the mainframe		7/8/2002		Table created.	
	7/12/2002	DHRM: Original server machine was having EJB exceptions during server startup and/or runtime		7/15/2002		A second machine was setup to test whether this was machine specific. The error is not reproduced on this second machine.	
	8/1/02	Availability of the UDDI server		On going		Availability of the UDDI server is still intermittent.	
	8/12/2002	DHRM: Mainframe validation, in addition to using zip code and city, also uses the area code of the phone number if it's stored in the test data.		8/16/2002		The way we decided to handle this is to not have phone numbers in the test data.	
12. PROTOCOL/SPECIFICATION ISSUES	Issues and/or difficulties specific to designing and coding for XML, SOAP, WSDL, and UDDI. NONE						
13. INTEROPERABILITY ISSUES	Discuss issues and/or difficulties that involve service-to-service interoperability between the various team applications. Called/Being Called						
	Service	VRS	DHRM	DMV	GMU	ABC	Description
	ChangeAddress – VRS	-	Y/Y	Y/Y	Y/Y	Y -/-	No Interoperability issues.
	GetAddresses(UserID) – VRS	-	Y/Y	Y/C-	C-/C-	Y/-	It seems that interoperability issues exist for this service. Tried several different ways of sending XML to both
	Batch GetAddresses – VRS	-	Y/Y		-/C		GMU said they were getting null value back from VRS. It would not be surprising if there were some issues
	ChangeAddress – DHRM	Y/Y	-	Y/Y	Y/Y	Y -/-	No Interoperability issues.
	GetAddresses(UserID)	Y/Y	-	-/C-	Y/C-	Y/-	Some SOAP errors that the other agencies received on their side that Mark Romiti was not able to reproduce

	– DHRM						on SilverStream side by calling service. For example, DHRM calling VRS was successful.
	Batch GetAddresses – DHRM	Y/Y	-		-/Y		GMU said that they called this service and got back rows of records.
<p>Legend:</p> <p>Y = Successful.</p> <p>Y- = tested a couple of times stand-alone but did not include in propagation because they are of type business addresses.</p> <p>C+ = Code and service generates successfully but got incorrect information.</p> <p>C- = Code generates successfully but caller gets SOAP errors.</p> <p>C = Code generates successfully but got null value back.</p> <p>- = Not aware of any info from other agencies or not tested.</p> <p>Successful responses are either from feedback from MS Developer Glenn H and GMU Developer Mahaela, calling getAddress for agencies I was successful, and a review of log data. Also, note in the case of DHRM even though other agencies got validation errors a majority of time I'm documenting this as successful for two reasons. First, Glenn and Mahaela were both able to successfully hit changeAddress service and get "true" responses back when using good data. Second, validation errors were data specific and mainframe related.</p> <p>Notes:</p> <ul style="list-style-type: none"> <li>• SilverStream developed interfaces that show web enabling of mainframes for DHRM and VRS – two separate agencies.</li> <li>• Validation issue with DHRM for city and zip code. If the proper city and zip code was not sent to the service, then an error message from mainframe is sent back to the requesting agency. So, several agencies thought the service wasn't working, when it was.</li> <li>• The WSDLs for DHRM and VRS had an error where the location field pointed to the localhost instead of the external IP. MS and GMU resolved this once we were in contact by giving them service urls. The corrected WSDL's were re-published.</li> <li>• Webmethods WSDL had interoperability issues initially. It was required that the WSDL be edited to remove the "wsdl" namespace that gets added by their tool to make it work. Once that was had massaged, then the services could be invoked.</li> <li>• In the case of the failures with getAddress, Mark thinks SOAP enablers for MS and GMU doesn't like whatever SilverStream is sending as a response. No errors occurred on SilverStream side (in terms of going between two SilverStream serviced sites), but errors did occurred on MS or GMU side. Not sure what the interoperability issue is.</li> </ul>							

14. "BEST PRACTICE" COMMENTS	<p>Technical or non-technical.</p> <ul style="list-style-type: none"><li>• All software practices should be adhered to when developing Web Services, especially in the areas of design.</li><li>• The Web Service, that is being registered, should be internally consumed before publishing the specs to the UDDI server.</li><li>• The XML going in or coming out of a Web Services should be verified against a schema.</li></ul>																																																						
15. OTHER CONCERNS / ISSUES	<p>Other team concerns/issues (technical, organizational, logistical or otherwise).</p> <p>SilverStream Web Serviced three functionalities for two agencies within three months. Since we had two agencies with mainframes versus the other vendors had one agency with no legacy integration, we believe we needed more time to thoroughly prove that Web Services can solve integration and interoperability pains. Even in the time-frame give, our engineer, Mark Romiti, was expert enough to show that there is a future in Web Services.</p> <p>Functionalities:</p> <ul style="list-style-type: none"><li>• ChangeAddress</li><li>• GetAddress(UserID)</li><li>• Batch GetAddress</li></ul> <p>Agencies:</p> <ul style="list-style-type: none"><li>• VRS</li><li>• DHRM</li></ul> <p>The developed interfaces showed real-time web-enablement of <b>mainframes (IBM and Unisys)</b> for DHRM and VRS.</p>																																																						
16. OTHER COMMENTS	<p>Comments on how the team project is going, successes, etc.</p> <p>Below is a timeline of when major tasks were started and completed:</p> <table><tr><th>Date Started</th><th>Task</th><th>Date Completed</th></tr><tr><td>6/6/2002</td><td>DHRM: Install SilverStream</td><td>6/7/2002</td></tr><tr><td>6/6/2002</td><td>VRS: Install SilverStream</td><td>6/10/2002</td></tr><tr><td>6/11/2002</td><td>Review of existing systems and interfaces has been done at both agencies</td><td>6/17/2002</td></tr><tr><td>6/11/2002</td><td>DHRM: Code UpdateAddress</td><td>6/18/2002</td></tr><tr><td>6/11/2002</td><td>DHRM: Code GetAddress(UserID)</td><td>6/17/2002</td></tr><tr><td>6/11/2002</td><td>DHRM: Code batch GetAddress</td><td>6/28/2002</td></tr><tr><td>6/11/2002</td><td>VRS: Code UpdateAddress</td><td>7/1/2002</td></tr><tr><td>6/11/2002</td><td>VRS: Code GetAddress(UserID)</td><td>6/14/2002</td></tr><tr><td>6/11/2002</td><td>VRS: Code batch GetAddress</td><td>7/1/2002</td></tr><tr><td>6/24/2002</td><td>DHRM: Web Service UpdateAddress</td><td>7/8/2002</td></tr><tr><td>6/24/2002</td><td>DHRM: Web Service GetAddress(UserID)</td><td>6/28/2002</td></tr><tr><td>6/24/2002</td><td>DHRM: Web Service batch GetAddress</td><td>7/15/2002</td></tr><tr><td>6/24/2002</td><td>VRS: Web Service UpdateAddress</td><td>7/15/2002</td></tr><tr><td>6/24/2002</td><td>VRS: Web Service GetAddress(UserID)</td><td>7/8/2002</td></tr><tr><td>6/24/2002</td><td>VRS: Web Service batch GetAddress</td><td>7/15/2002</td></tr><tr><td>6/24/2002</td><td>DHRM &amp; VRS: Create the Log table</td><td>6/25/2002</td></tr><tr><td>6/24/2002</td><td>DHRM &amp; VRS: Create the COTS User table</td><td>6/25/2002</td></tr></table>	Date Started	Task	Date Completed	6/6/2002	DHRM: Install SilverStream	6/7/2002	6/6/2002	VRS: Install SilverStream	6/10/2002	6/11/2002	Review of existing systems and interfaces has been done at both agencies	6/17/2002	6/11/2002	DHRM: Code UpdateAddress	6/18/2002	6/11/2002	DHRM: Code GetAddress(UserID)	6/17/2002	6/11/2002	DHRM: Code batch GetAddress	6/28/2002	6/11/2002	VRS: Code UpdateAddress	7/1/2002	6/11/2002	VRS: Code GetAddress(UserID)	6/14/2002	6/11/2002	VRS: Code batch GetAddress	7/1/2002	6/24/2002	DHRM: Web Service UpdateAddress	7/8/2002	6/24/2002	DHRM: Web Service GetAddress(UserID)	6/28/2002	6/24/2002	DHRM: Web Service batch GetAddress	7/15/2002	6/24/2002	VRS: Web Service UpdateAddress	7/15/2002	6/24/2002	VRS: Web Service GetAddress(UserID)	7/8/2002	6/24/2002	VRS: Web Service batch GetAddress	7/15/2002	6/24/2002	DHRM & VRS: Create the Log table	6/25/2002	6/24/2002	DHRM & VRS: Create the COTS User table	6/25/2002
Date Started	Task	Date Completed																																																					
6/6/2002	DHRM: Install SilverStream	6/7/2002																																																					
6/6/2002	VRS: Install SilverStream	6/10/2002																																																					
6/11/2002	Review of existing systems and interfaces has been done at both agencies	6/17/2002																																																					
6/11/2002	DHRM: Code UpdateAddress	6/18/2002																																																					
6/11/2002	DHRM: Code GetAddress(UserID)	6/17/2002																																																					
6/11/2002	DHRM: Code batch GetAddress	6/28/2002																																																					
6/11/2002	VRS: Code UpdateAddress	7/1/2002																																																					
6/11/2002	VRS: Code GetAddress(UserID)	6/14/2002																																																					
6/11/2002	VRS: Code batch GetAddress	7/1/2002																																																					
6/24/2002	DHRM: Web Service UpdateAddress	7/8/2002																																																					
6/24/2002	DHRM: Web Service GetAddress(UserID)	6/28/2002																																																					
6/24/2002	DHRM: Web Service batch GetAddress	7/15/2002																																																					
6/24/2002	VRS: Web Service UpdateAddress	7/15/2002																																																					
6/24/2002	VRS: Web Service GetAddress(UserID)	7/8/2002																																																					
6/24/2002	VRS: Web Service batch GetAddress	7/15/2002																																																					
6/24/2002	DHRM & VRS: Create the Log table	6/25/2002																																																					
6/24/2002	DHRM & VRS: Create the COTS User table	6/25/2002																																																					

	7/19/2002	DHRM: Test Web Service UpdateAddress	8/16/2002
	7/19/2002	DHRM: Test Web Service GetAddress(UserID)	8/16/2002
	7/19/2002	DHRM: Test Web Service batch GetAddress	8/16/2002
	7/19/2002	VRS: Test Web Service UpdateAddress	8/16/2002
	7/19/2002	VRS: Test Web Service GetAddress(UserID)	8/16/2002
	7/19/2002	VRS: Test Web Service batch GetAddress	8/16/2002
	7/22/2002	DHRM & VRS: Create the JSPs for the user interface	8/7/2002
	8/1/2002	DHRM: Publish Web Service UpdateAddress	8/6/2002
	8/1/2002	DHRM: Publish Web Service GetAddress(UserID)	8/6/2002
	8/1/2002	DHRM: Publish Web Service batch GetAddress	8/6/2002
	8/1/2002	VRS: Publish Web Service UpdateAddress	8/6/2002
	8/1/2002	VRS: Publish Web Service GetAddress(UserID)	8/6/2002
	8/1/2002	VRS: Publish Web Service batch GetAddress	8/6/2002
	8/6/2002	DHRM: Call other Agencies Web Service UpdateAddress	8/16/2002
	8/6/2002	DHRM: Call other Agencies Web Service GetAddress(UserID)	8/16/2002
	8/6/2002	DHRM: Call other Agencies Web Service batch GetAddress	8/16/2002
	8/6/2002	VRS: Call other Agencies Web Service UpdateAddress	8/16/2002
	8/6/2002	VRS: Call other Agencies Web Service GetAddress(UserID)	8/16/2002
	8/6/2002	VRS: Call other Agencies Web Service batch GetAddress	8/16/2002
<b>17. COST/TIME ESTIMATE</b>	Track the time the full team spends for 1) Meetings, 2) Development, 3) Training, and 4) Testing.		
	<b>Task</b>	<b>Duration</b>	<b>Time Expended To Date (hrs)</b>
	<b>Meetings</b>	4/25/2002 - 8/5/2002	16
		4/25/2002 - 8/19/2003	20
		4/25/2002 - 8/19/2004	24
	<b>Total Meetings</b>		<b>60</b>
	<b>Development</b>	6/10/2002 - 7/19/2002	600
		6/24/2002 - 7/19/2002	40
	<b>Total Development</b>		<b>640</b>
	<b>Testing</b>	7/22/2002 – 8/19/2002	80
	<b>Total Testing</b>		<b>80</b>
	<b>Training</b>	8/1/2002 - 8/19/2002	20
	<b>Total Training</b>		<b>20</b>
	<b>Total Hours</b>		<b>800</b>

## **Appendix I: Team 4 – Team Tortoise Final Report – City of Virginia Beach, Roanoke County & Sun Micorssystems**

### **1. Project Plan (Schedule)**

<b>Meeting</b>	<b>Date</b>	<b>Location</b>	<b>% Complete</b>
Kick Off Meeting	3/28/2002	VRS	100
COTS Status Meeting	4/25/2002	VRS	100
Team Meeting	5/24/2002	Conference Call	100
COTS Status Meeting	5/30/2002	VRS	100
COTS Status Meeting	6/20/2002	VRS	100
Team Meeting	6/21/2002	Virginia Beach	100
Training (Tutorial based)	6/24/2002	Virginia Beach, Roanoke	100
Development	7/8/2002	T4 Consulting Group/Sun	75
Testing	7/29/2002	T4 Consulting/Sun	20
COTS Status Meeting	7/18/2002	VRS	100
Final COTS Status Meeting	8/19/2002	VRS	100

### **2. List of Team Name, Members, Roles and Responsibilities**

#### **City of Virginia Beach**

Joanne Pilcher, Agency Representative, Project Support  
Andrea Jamison, Developer (removed after May 28, 2002)  
Andy Lam, Developer (brought in after May 28, 2002)

#### **Roanoke County**

Elton Ghee, Agency Representative, Project Support  
Nicole Bird, Developer (brought in after 5/24/2002)

#### **Sun Microsystems**

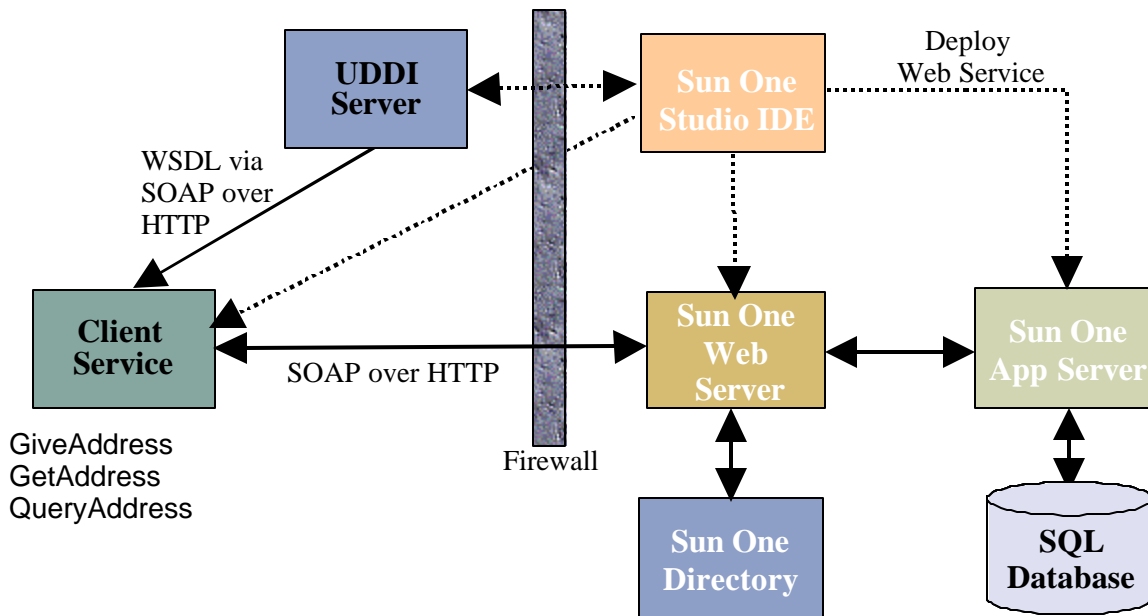
Wayne Cox, Sales Engineer/Support  
David Wilkinson, Sales Representative  
Jake Martin, Software Sales Representative  
Eric Reed, Software Sales Engineer  
Alexi Polenur, Consulting Developer (brought in after 7/18/2002)  
Sekharam Kasturi, Consulting Developer (brought in after 7/18/2002)

#### **T4 Consulting Group (T4CG)**

Paul Hoehne, Developer (brought in after 6/20/2002)  
Brian Field, Developer (brought in after 6/20/2002)  
Bob Lytle, CTO/Support (brought in after 6/20/2002)

### 3. Proof-of-Concept Design

## Proof of Concept Design



## 4. Business/Functional Requirements

**Business Requirement:** To implement the web services change of address proof-of-concept representing the City of Virginia Beach and Roanoke County using the Sun One product suite. Get initial communications occurring between the localities, and then distribute change of address requests to the other COTS teams.

## 5. Project Equipment Description

Hardware and software is hosted by T4 Consulting Group, in Leesburg, Virginia and provided by Sun Microsystems.

### Hardware Platform

Sun Solaris Enterprise 220R server with two 450MHz  
UltraSPARC-II processor, 4MB E-cache, 1GB memory, two 36GB disks

### Software Platform

Sun One Web Server  
Sun One Application Server  
Sun One Studio (Java IDE)  
Sun One Directory Server  
Pointbase Database (freeware)

## 6. Additional Acquisition and Installation Activities

Install Sun One software suite at T4 Consulting Group. Also, install Sun One Studio on workstation at localities for tutorial training.

## 7. Server Requirements Changes

The original plan was to have individual servers installed in Virginia Beach and Roanoke. Since the localities did not have servers readily available, Sun agreed to provide these servers. The first server was setup, delivered, and sent to Roanoke. However, this server experienced an accidental failure and was returned to Sun. Because the team was behind schedule, Sun revised the plan and decided to host the server and software at T4 Consulting Group.

## 8. Networking Requirements Changes

The firewall had to be re-configured to allow access to the Server where the web services reside. Additionally, ports were opened to access the application server, database, and web server.

## 9. Training Requirements

The team received training by

- Self study on Web Services concepts
- Reading the Introduction to Web Services whitepaper (provided by Sun)
- Going through the Sun One Studio Tutorial (formerly Forte)
- Conference calls
- One-on-one consultation calls.

## 10. Testing Plan

Test plan not completed.

## 11. Development/Technical

This team experienced installation and configuration issues in regards to the releases of the software, patches, and updates to the separate components. Obtaining the “right” development resources to support the project proved to be difficult.

Initially, we started with the newly released version of Java IDE that included the latest web services protocol and APIs. After installation issues were encountered, the decision was made to go to the previous version the developer was familiar with. The web services APIs were an additional update module to the previous version that had to be installed.

During the early stages of development the IDE generated invalid SOAP code. This error may have been caused by the decision to use a previous version and adding the API modules. To overcome this issue, the developer used open source tools to generate the SOAP code.

Generally, software engineers are well versed in one aspect of the overall solution (i.e., portal server, application server, directory) and finding one available technical person who understood and knew how to integrate all the various components proved to be a downfall for the team.

The project schedule also became a challenge for the team. The final development phase of the project occurred during the busiest time of year for Sun. Today's economic environment being as it is, it was difficult to get developers off revenue generating projects. As a result, we had to use the developers that were available at any given time. Each new developer required a ramp up time to get familiar with the project and its goals. This hurdle proved to be impossible to overcome. Additionally, the summer schedule interfered with the availability of resources, since Sun shuts down during the first week of July.



## 12. Protocol/Specification Issues

The only protocol issue experience by the team was the WSDL was supposed to be the same for all teams and they were not.

## 13. Interoperability Issues

The team experienced interoperability issues with the integration of the web server and the application server where the application server stopped communicating with the web server. This issue caused major delay during the most critical time of the project.

## 14. “Best Practice” Comments

- A web services effort needs to be carefully managed as a development project. As an integration effort, web services require skilled project management to insure the proper planning, communication (especially among the stakeholders), and development guidelines (detailed requirements, specifications, security issues).
- Web services development requires a variety of skilled and experienced technologists – analysts, system engineers, network and security engineers, and web experienced programmers.

## 15. Other Concerns/Issues

A Web services project requires central controls and policy decisions to make sure all the stakeholders understand the rules to achieve the success of the web service. While this project was just a proof-of-concept, the real world security issues would need to be resolved to the satisfaction of the Security Officer before a web services project could proceed.

## 16. Other Comments

Although the team was unable to successfully complete the development of the web service proof-of-concept, the following goals of the team were achieved:

- Understand the concepts, issues, process, and effort involved in a Web Services project.
- Discover how/if web services could benefit our business and technology issues (is it a viable integration solution or just hype?).
- Document a lessons-learned approach for venturing into the web services technology arena.

## 17. Cost/Time Estimate

The following cost/time estimates reflect the total number of hours put in collectively by all individuals on the team.

1. Meetings, 178 hours
2. Development, 120 hours
3. Training, 92.5 hours
4. Testing, 20 hours

## Appendix J: Team 5 – *Mad Dogs* Final Report – DOE, DMAS & Software AG

### 1. Project Plan (Schedule)

High-level project plan with list of meetings, time and location giving percent complete.

Task/Milestone	Comp Date	Responsible	Status
Server is set up and available	07-Jun	SAG	complete
Obtain Final schema information	07-Jun	BEA	received
WSDL information is complete	07-Jun	John	complete
Access information provided to team	10-Jun	SAG	complete
Obtain printed documentation(book)	10-Jun	all	complete
Obtain online documentation	10-Jun	John	complete
Training Overview	12-Jun	all	complete
Generate DDL - DMAS Table	14-Jun	Henry/Dave	complete
Generate DDL - DOE Table	14-Jun	Dave/Henry	complete
Create DMAS Table	14-Jun	Dave/Henry/John	complete
Create DOE Table	14-Jun	Dave/Henry/John	complete
Set up Conference Calls	18-Jun	John	complete
Team meeting	19-Jun	all	complete
WSWG meeting	20-Jun	all	complete
Create Stored Procedure - VDOE Retrieve	21-Jun	John/Dave/Henry	complete
Create Stored Procedure - VDOE Update	21-Jun	John/Dave/Henry	complete
Create Stored Procedure - DMAS Retrieve	21-Jun	John/Dave/Henry	complete
Create Stored Procedure DBAS Update	21-Jun	John/Dave/Henry	complete
Team Meeting	26-Jun	all	complete
Create Web Service Servlet Inquire	28-Jun	Dave/Henry/John	complete
Create Web Service Servlet Change	28-Jun	Dave/Henry/John	complete
Create Web Service Servlet - List Changes	28-Jun	Dave/Henry/John	complete
Create Mediator Sequence to update appropriate agency database	28-Jun	Dave/Henry/John	complete
create portal (HTML)	28-Jun	Dave/Henry/John	complete
Create Address Change Push Service	28-Jun	Dave/Henry/John	complete

Begin Testing	01-Jul Dave/Henry/John	complete
Load Test Data in DMAS Table	01-Jul Henry/Dave	complete
Load Test Data in DOE Table	01-Jul Dave/Henry	complete
Unit test	16-Jul John/Dave/Henry	complete

Task/Milestone	Comp Date	Responsible	Status
Begin Interoperability Testing	19 Jul	all with other teams	Started Aug 12
Team Meeting	10-Jul	all	Complete
Complete Testing	12-Jul	Dave/Henry/John	Incomplete
Team Meeting	17-Jul	all	Complete
WSWG Meeting	18-Jul	all	Complete
Team Meeting	24-Jul	all	starts 11 a.m.
Team Meeting	31-Jul	all	starts 11 a.m.
Team Calls	When needed		As needed
Interoperability Testing Ends	01-Aug	all with other teams	
WSWG Meeting	19-Aug	all	Complete
Project Complete	-Aug	All	Complete
Assist in Report Development	after 5-Aug	Nelly/Bethann	Complete
Track Hours in Meetings	ongoing	all	Complete
UDDI Registry Complete		VIPNET	Complete
Register our service w/VIPNET		all	Complete
Create Test Plan		Tim Bass	Complete

## 2. List of Team Name, Members, Roles and Responsibilities

Statement of team member roles and responsibilities – who is doing what. Highlight new/additional team members being brought in to assist since last report.

### Virginia Department of Education

Bethann Canada, Agency Representative Co-Team Lead/Technical Support  
David Hanzlik, Advisor/Developer

### Virginia Department of Medical Assistance Services

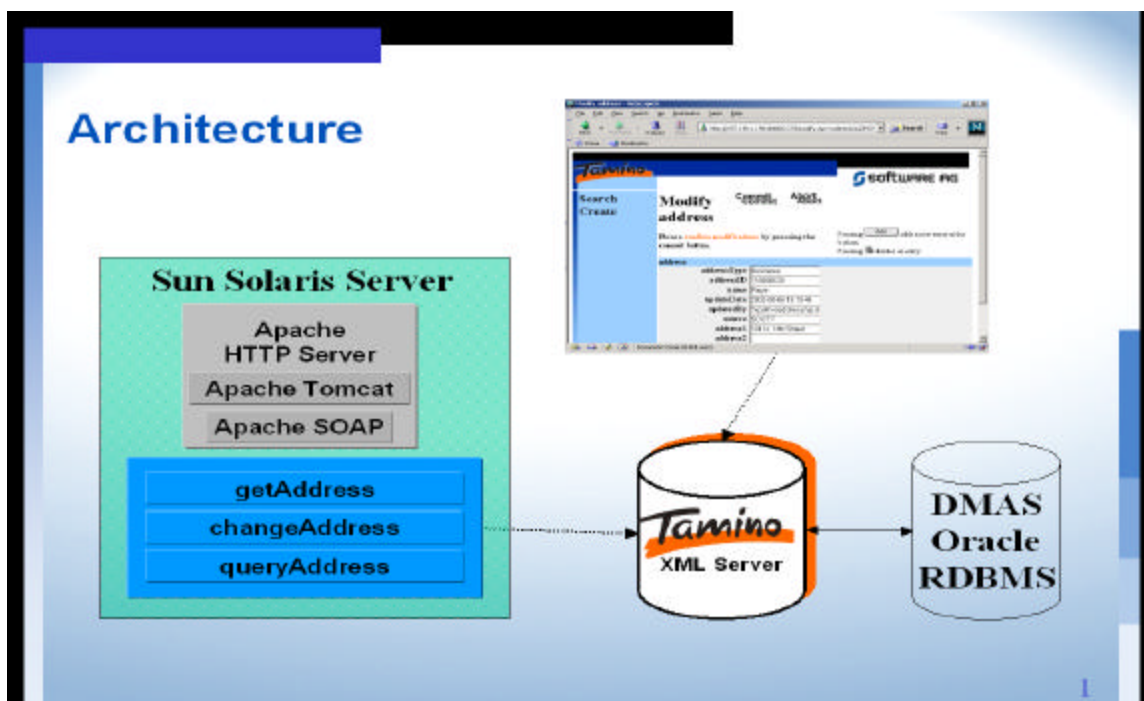
Nelly Romero, Agency Representative/Report Administrator  
Henry Witz, Advisor/Developer

### Software AG, Inc.

Sari Clark, Account Executive/Support  
Eric Wood, Sales Engineer/Support  
John Taylor, IT Architect, Co-Team Lead/Developer  
Joel Patterson, Sales Engineer/Support  
Trevor Ford, Research and Development Specialist

## 3. Proof-of-Concept Design

Attach general design of your application narrative, flow diagrams, etc. – whatever you think will adequately convey the team's approach in terms of design.



## 4. Business/Functional Requirements

Statement of the specific business/functional requirements that are driving the team's application design.

Implement the Web Services Change of Address Proof-Of-Concept representing the Department of Education and the Department of Medical Assistance Services for two types of Addresses:  
Home and Street, and  
Business

Represent the Web Service "ennoblement" of DOE and DMAS systems currently residing in Oracle DBs  
Implement the distribution of Change of Address requests to the other teams participating in the Web Services Proof-Of-Concept

## 5. Project Equipment Description

Full description of the hardware and software to be used, who is providing it and where it is located.

**Hardware and software is hosted by Software AG, Inc. in Reston, Virginia**

### **Hardware Platform**

Sun Solaris 2.7

### **Software Platform**

Apache HTTPS Server 1.3

Apache Jakarta Tomcat Servlet & JSP Engine 3.3

Apache SOAP 2.3

Software AG Tamino XML Server (including X-Node ODBC) 3.1

Software AG EntireX XML Mediator 7.1

### **Agency Representative Software**

Oracle RDBMS 8.1

## 6. Additional Acquisition and Installation Activities

Indicate whether additional hardware and software were purchased or acquired for the proof-of-concept.

Additional software will be acquired such as freeware development tools and will be listed here as obtained.

## 7. Server Requirements Changes

Describe changes to existing server or acquisition that will be needed before testing can begin. Describe the factors that influenced your decisions.

Configured Apache Server for HTTPS and SSL. But we never utilized any secure communications.

## 8. Networking Requirements Changes

Describe changes to existing network or procurement that will be needed before testing can begin. Describe the factors that influenced your decisions.

Re-configured firewall to allow access to the Tomcat Server where the web services resides. Due to Software AG's network security policies, we had to clarify the use of the server for IT and compromise on server availability. We opened all necessary ports, but made the box available on during the hours of 7 AM and 7 PM.

## 9. Training Requirements

Identify the technical and business training that has been or will be needed to prepare staff for proof-of-concept project.

Received two hours high level training June 12, 2002, covering the following points:

- General overview of Web Services
- General overview of Software AG's products

Each developer also received a book titled XML and Web Services Unleashed SAMS. Each team member will track time spend studying/training for our proof-of-concept.

June 26, 2002 Software AG provided DOE and DMAS technical staff with overview of architecture and drilled down on each of the components.

July 24, 2002 Software AG, DOE and DMAS technical staff conducted a 3-hour, training/development/testing session. Software AG provided a review of all application software and discussed various features and limitations of the product at that time. The underlying concepts of Web Services were re-capped in light of discussion at the July 17th workgroup meeting.

The majority of the training required to implement the POC has been in the form of on-the-job self-learning and consultation with experienced developers reiterating the nicety of having someone on the project whom "has been there and done that".

## 10. Testing

Reference No.	Test Item	Module Name	Action	Expected Results
1.1	ddl		Create Table	Oracle table is created in the appropriate schema. Describe table_name should show the correct table format.
2.1	SQL		Load test data into table	Expected number of rows should be loaded into table.
3.1	Stored Procedure		Retrieve one address from table	When provided with unique ID, and address type, successful return of address. Status of record not found when address is not on file.
4.1	getAddress Servlet		Retrieve address from other Systems	Successfully retrieve an address from other systems.
4.2	changeAddress Servlet		Change address in other Systems	Successfully update addresses in other systems
4.3	queryAddress Servlet		List changed addresses from other systems	Successful retrieval of addresses that we've changed in other systems.
5.1	Portal		Web Site for project	Web site is accessible on the Internet.
6.1	Push Service		Propagate addresses to other systems	Successfully provide addresses to subscribers. Log entries confirm propagation and receipt.
7.1	SSL		Web server is configured for SSL.	Verify SSL connection.
7.2	IP filtering		IP filtering is in place.	Verify IP Filtering.
8.1	UDDI entry		UDDI entries are properly inserted.	Verify UDDI entries.



## Test Results

### Item 1.1

The following Data Definition Language (DDL) script was used to create the Oracle table in the scott schema:

```
CREATE TABLE PERSON_ADDRESS (  
  ADDRESSID      VARCHAR2 (9)  NOT NULL,  
  FIRST_NAME     VARCHAR2 (30),  
  MIDDLE_INIT    VARCHAR2 (1),  
  LAST_NAME      VARCHAR2 (30) NOT NULL,  
  ADDRESSTYPE    VARCHAR2 (20) NOT NULL,  
  ADDRESS1       VARCHAR2 (30),  
  ADDRESS2       VARCHAR2 (30),  
  CITY           VARCHAR2 (30) NOT NULL,  
  STATE          VARCHAR2 (2)  NOT NULL,  
  ZIP5           VARCHAR2 (5)  NOT NULL,  
  ZIP4           VARCHAR2 (4),  
  LAST_UPDATED_BY VARCHAR2 (500),  
  LAST_UPDATE_DATE DATE,  
  CONSTRAINT PERSON_ADDRESS_PK  
  PRIMARY KEY ( ADDRESSID, ADDRESSTYPE ));
```

Execution produced the following results:

```
SQL> show user
```

```
USER is "SCOTT"
```

```
SQL> descr person_address
```

Name	Null?	Type
ADDRESSID	NOT NULL	VARCHAR2(9)
FIRST_NAME		VARCHAR2(30)
MIDDLE_INIT		VARCHAR2(1)
LAST_NAME	NOT NULL	VARCHAR2(30)
ADDRESSTYPE	NOT NULL	VARCHAR2(20)
ADDRESS1		VARCHAR2(30)
ADDRESS2		VARCHAR2(30)
CITY	NOT NULL	VARCHAR2(30)
STATE	NOT NULL	VARCHAR2(2)
ZIP5	NOT NULL	VARCHAR2(5)
ZIP4		VARCHAR2(4)
LAST_UPDATED_BY		VARCHAR2(500)
LAST_UPDATE_DATE		DATE

## Item 2.1

The data was initially loaded into Oracle using the two separate SQL files whose line counts are shown below:

```
usrssun10:/usr/ORACLE> wc -l data*.sql
  250 data5.sql
 1001 data7rev.sql
 1251 total
```

data7rev.sql contains an additional line which is a delete statement to eliminate records not of address type Home or Business.

When the data was reloaded through Tamino, the deletion of address types other than Home or Business was not included. Therefore, the total number of records in the database is 1250. This was done in order for Tamino to update the Oracle database, it must establish a key to each record in the database. This key is the "INO number". We established the INO data by deleting the existing database data and reloading it through Tamino. This was the quickest easiest solution for us but is something other potential Tamino users may need to be aware of as a conversion issue.

```
select count(*) from person_address
SQL> /
```

```
COUNT(*)
```

```
-----
      1250
```

```
SQL>
```

## Item 3.1

The stored procedure was not needed since the X-node product links directly to Oracle.

## Item 4.1

Successfully developed and unit-tested the **getAddress** service.

## Item 4.2

Successfully developed and unit-tested the **changeAddress** service.

### Item 4.3

Successfully developed and unit-tested the **queryAddress** service.

### Item 5.1

A portal was generated using Tamino X-Application, a free tool available through the Tamino Developer Community. This allows basic Create, Retrieve, Update, and Delete functions. We used this tool for internal and unit tests.

Modify address - Netscape 6

File Edit View Search Go Bookmarks Tasks Help

Back Forward Reload Stop <http://157.189.11.90:8080/COTS/modify.jsp?coded=63225937> Search Print

Home Bookmarks

**Tamino** **SOFTWARE AG**

**Modify address** **Commit** **Abort**

Please **confirm modifications** by pressing the **commit** button.

Pressing  adds a new entry at the bottom.  
Pressing deletes an entry.

address	
addressType	Business
addressID	118000630
name	Raye
updateDate	2002-08-06 15:19:46
updatedBy	?xpath=/address/upd
source	SCOTT
address1	101 N. 14th Street
address2	

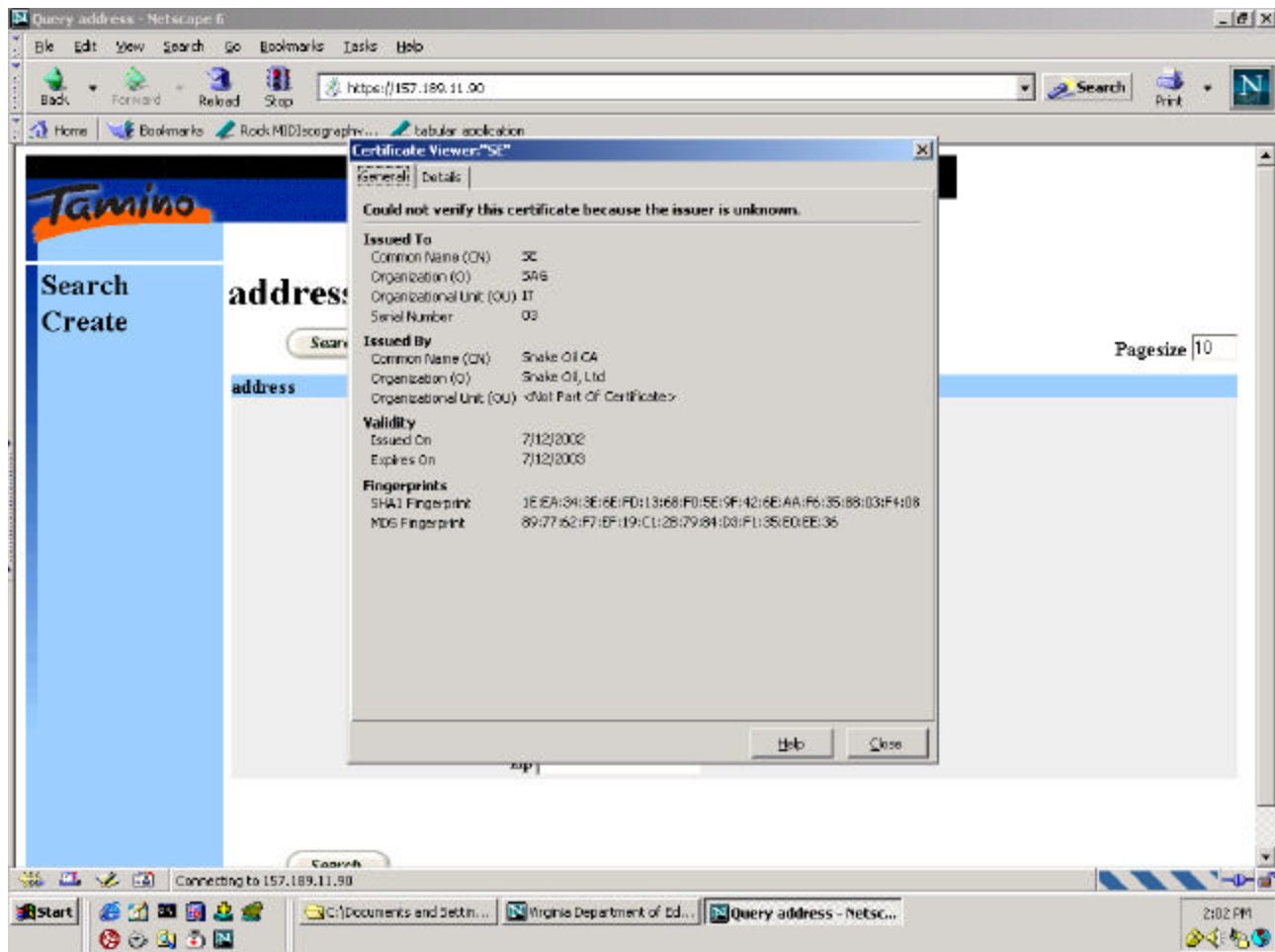
Document: Done (0.861 secs)

### Item 6.1

Successfully developed and partially unit-tested the **pushAddress** service.

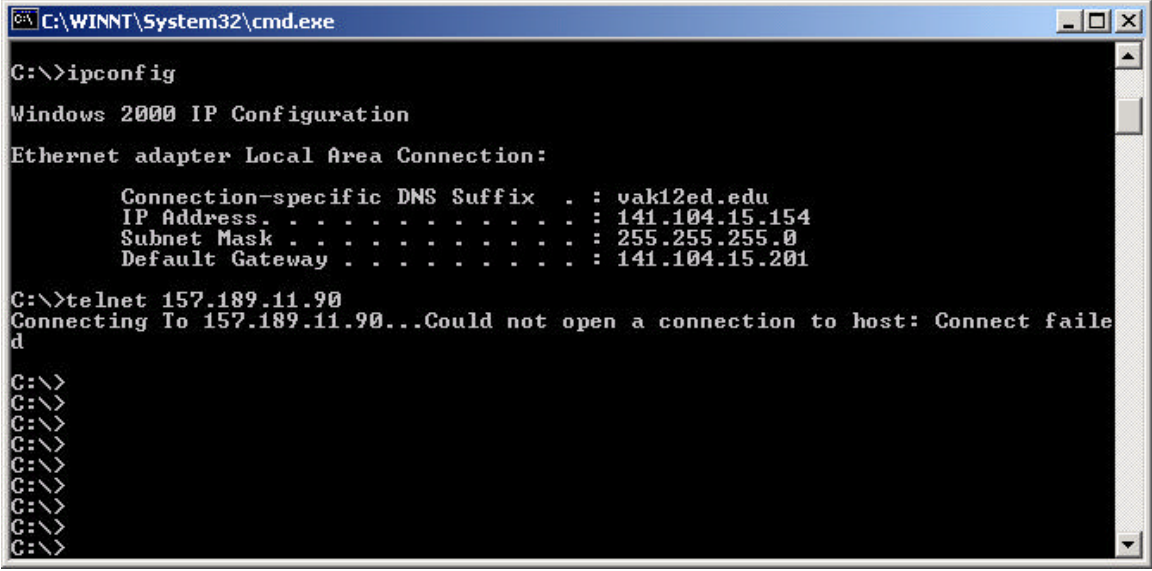
## Item 7.1

SSL has been installed on the server as evidenced by the following request for certificate verification when accessing the server using https protocol. For the purposes of the Proof-of-concept, testing certificates from Snake Oil CA have been used.



## Item 7.2

IP filtering has been put in place on the Software AG firewall so that only developers accessing the server from an authorized IP address (or inside the Software AG network) are able to affect changes to the application software. Anyone else attempting to telnet to the server will receive an error message similar to the one shown below.



```
C:\WINNT\System32\cmd.exe

C:\>ipconfig

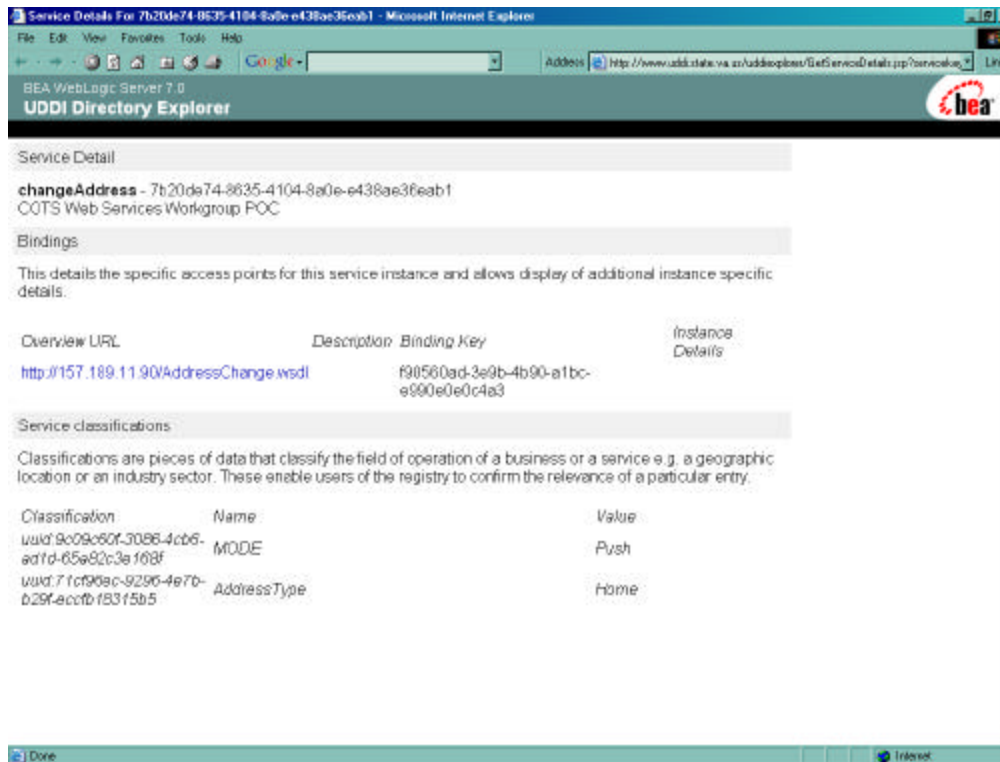
Windows 2000 IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : vak12ed.edu
    IP Address. . . . . : 141.104.15.154
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 141.104.15.201

C:\>telnet 157.189.11.90
Connecting To 157.189.11.90...Could not open a connection to host: Connect failed
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
```

## Item 8.1



## 11. Development/Technical

High-level description of specific development/technical issues and/or difficulties encountered during coding.

Configuration and integration of various components resulted in some delays due to incomplete or poor documentation and the learning curve:

- The Apache Tomcat software (free software) documentation was lacking, error messages were cryptic, and there was no technical support available.
- The team experienced difficulty in establishing a link from Tamino to the Oracle database.
- Several configuration changes to the HTTP server required firewall changes and subsequent re-boots that had to take place on weekends or after hours.
- Configuring the HTTP server for SSL was not in the team's original plan, but did not result in a significant delay.

## 12. Protocol/Specification Issues

Issues and/or difficulties specific to designing and coding for XML, SOAP, WSDL, and UDDI.

- XML – no issues experienced.
- SOAP – no issues experienced.
- WSDL – initially required some minor editing. Various teams needed to come to agreement on issues such as naming conventions. In the end, there were multiple conflicting versions of the WSDL. Versus the single standard interface that was initially agreed upon.
- UDDI – Significant issues that all teams struggled with resulted in delays. Decision to proceed using static addresses resulted in unanticipated code changes. Ultimately, we did not use the UDDI registry as part of the project.

## 13. Interoperability Issues

Discuss issues and/or difficulties that involve service-to-service interoperability between the various team applications.

- Inter-team interoperability testing was successful.
- We successfully delivered all of our services and the WSDL describing them.
- We successfully pushed an address change to one other team.
- The other teams were able to consume our WSDL.
- We were NOT included in the test harness before project completion. Therefore, we were not able to complete our interoperability testing.

## 14. Other Concerns/Issues

Other team concerns/issues (technical, organizational, logistical or otherwise).

- Software AG's security policies caused interruptions in the availability of the server on the Internet.
- Test data needed cleaning prior to use.

## 15. Other Comments

Comments on how the team project is going, successes, etc.

- Although the team was behind schedule at several intervals, each member made time for conference calls (sometimes daily) to bring the project back on track.
- Significant effort on the part of the team's vendor technical staff should be recognized as well as the effort by BEA/VIPNet to provide UDDI and testing harness.
- The team wishes to recognize the significant commitment and contributions of all members in this volunteer proof-of-concept.

## Lessons Learned

- If free software is utilized:
  - Need to have access to experienced resources.
  - Need to provide the necessary training to staff.
  - Consider buying support from reseller.
- The tools are maturing in terms of Web Services development but are not quite there for testing.
- Interoperability requires significant coordination. One of the major hindrances to interoperability was the lack of detailed specifications. And that despite our best efforts, adherence to the interfaces agreed upon were not consistent.
- Web Services are technically ready for implementation. As with any new technology, maturity levels could impact some areas of development. But overall, the technology and the paradigm work.
- UDDI as a directory service for web services, is complex. Grasping a complete understanding of WSDL and UDDI can be difficult. Unless public directories are a definite requirement, we would recommend taking the approach the workgroup ended with – to statically bind the services.



## 16. Cost/Time Estimate

Track the time the full team spends for 1) Meetings, 2) Development, 3) Training, and 4) Testing.

<b>Description of Activity</b>	<b>Bethann</b>	<b>Dave</b>	<b>Nelly</b>	<b>Henry</b>	<b>John</b>	<b>SAG</b>	<b>Total</b>
Meetings	32	20	26	22	46	16	<b>158</b>
Development/unit test	0	9	0	2	204	56	<b>271</b>
Test Data (generating/loading)	0	16		10	15		<b>41</b>
Training/Reading	5	6	5	6	12		<b>34</b>
Management (e-mails, report, etc)	12	2	22	2	8		<b>46</b>
Integration Testing	1						<b>1</b>
Install/Config/SW/Ntwk						40	<b>40</b>
<b>Grand Total</b>	<b>50</b>	<b>53</b>	<b>53</b>	<b>42</b>	<b>281</b>	<b>112</b>	<b>579</b>

## Appendix K: Team 6 Final Report – GMU & webMethods

<b>1. PROJECT PLAN (SCHEDULE)</b>	Meeting	Date	Location	Pct Compl
	Kickoff	03/28	VRS	100%
	COTS Status Meeting	04/25	VRS	100%
	Team Meeting	05/10	webMethods	100%
	Acquire HW/SW	05/15		100%
	Team Meeting	05/23	webMethods	100%
	COTS Status Meeting	05/30	VRS	100%
	Development Start	06/10		100%
	Team Meeting	06/17	webMethods	100%
	Team Meeting	06/19	GMU	100%
	COTS Status Meeting	06/20	VRS	100%
	Team Meeting	06/27	GMU	N/A
	Team Meeting	07/01	GMU (2 hrs)	100%
	Test Plans Compl	07/08		100%
	Team Meeting	07/12	GMU (2 hrs)	100%
	Team Meeting	07/16	GMU (2 hrs)	100%
	Training	07/16		N/A
	COTS Status Mtg	07/18		100%
	Integration Test Start	07/21		100%
	Team Meeting	07/23	GMU (2 hrs)	100%
	Development End	07/26		100%
	Team Meeting	07/30	GMU (2 hrs)	100%
	Integration Test End	08/16		100%
	Group Presentation	08/19		100%
	Final Report Start	08/20		
	Final Report End	09/10		
<b>2. LIST OF TEAM NAME, MEMBERS, ROLES AND RESPONSIBILITIES</b>	Company	Name	Responsibilities	
	webMethods	Chris Demory	Project Support <ul style="list-style-type: none"> <li>Engineering Resources</li> </ul>	

	webMethods	Jaime Moore	<ul style="list-style-type: none"> <li>• Training/Technical Support</li> </ul> Project Support
	webMethods	Floyd West	<ul style="list-style-type: none"> <li>• Engineering Resources</li> <li>• Training/Technical Support</li> </ul> Vendor Technical Lead
	GMU	John Creuziger	<ul style="list-style-type: none"> <li>• App Design</li> <li>• App Development</li> </ul> Project Mgmt Project Support
	GMU	Mihaela Enache	Agency Technical Lead <ul style="list-style-type: none"> <li>• App Design</li> <li>• App Development</li> </ul>
<b>3. PROOF-OF-CONCEPT DESIGN</b>	See attachment 1.		
<b>4. BUSINESS/FUNCTIONAL REQUIREMENTS</b>	Business Requirements <ul style="list-style-type: none"> <li>• Complete proof-of-concept address change application</li> <li>• Demonstrate the feasibility of integrating applications with SCT Banner</li> <li>• Realistically determine staff and training investments</li> <li>• Work within the computing architecture of GMU</li> </ul> Technical Requirements <ul style="list-style-type: none"> <li>• Use webMethods products to develop/host application</li> <li>• Use XML, SOAP, WSDL, and UDDI</li> <li>• Use Java2 technology suite</li> <li>• Use SSL</li> <li>• Use Oracle Database</li> </ul> Functional Requirements <ul style="list-style-type: none"> <li>• Authenticate requestor via ID/PIN combo</li> <li>• Prompt for pertinent address type(s)</li> <li>• Display old address info/prompt for new</li> <li>• Update appropriate local addresses</li> <li>• Route request to each development team</li> <li>• Process web services updates from other teams</li> <li>• Log all address change activity (or lack)</li> <li>• Generate confirmation to requestor</li> </ul>		

<b>5. PROJECT EQUIPMENT DESCRIPTION</b>	<p>Hardware (Provided by GMU and located at GMU)</p> <ul style="list-style-type: none"> <li>• Gateway Pentium IV <ul style="list-style-type: none"> <li>➤ 1.9 Ghz 4 CPU</li> <li>➤ 512 MB RAM/40 GB Storage</li> <li>➤ Windows XP Professional</li> </ul> </li> <li>• Sun E-420R Ultra-80 <ul style="list-style-type: none"> <li>➤ 4/450 Mhz CPUs</li> <li>➤ 4 GB RAM/1 TB Storage</li> <li>➤ Solaris 2.8</li> </ul> </li> </ul> <p>Software (Provided by webMethods and installed at GMU)</p> <ul style="list-style-type: none"> <li>• webMethods Integration Server</li> <li>• webMethods Developer</li> <li>• webMethods Enterprise Server</li> </ul> <p>Software (Provided by GMU and installed at GMU)</p> <ul style="list-style-type: none"> <li>• Oracle 8i</li> </ul>
<b>6. ACQUISITION AND INSTALLATION ACTIVITIES</b>	<p>webMethods Integration Server, Developer, Business Process Modeling, Trading Networks, and Workflow software acquired and installed (no cost to agency). Updated trial license keys on 6/18 and 7/16.</p>
<b>7. SERVER REQUIREMENTS</b>	<p>No server changes or acquisitions required</p>
<b>8. NETWORKING REQUIREMENTS</b>	<p>Static IP address obtained and DNS entries established 7/16</p>
<b>9. TRAINING REQUIREMENTS</b>	<p>Although a developer training class was repeatedly offered by webMethods, scheduling conflicts prohibited attendance. However, lack of training was not a factor in the completion of the proof-of-concept project.</p>
<b>10. TESTING PLAN</b>	<p>See attachment 2. Unit testing was successfully completed for all services – each service worked as expected when GMU executed them internally. Actual programming logic that was implemented was slightly different, but the major steps were identical.</p>
<b>11. DEVELOPMENT/TECHNICAL</b>	<p>Minor problem noted 07/18 consuming internal services was resolved by the installation of a webMethods Service Pack. Overall, no major technical issues surfaced.</p> <p>Installation and configuration activities were relatively easy and required only minimal assistance from the vendor. Programmers were already somewhat familiar with underlying protocols, so they presented little difficulty.</p> <p>The integrated, graphical webMethods development environment resulted in minimal Java coding and built-in services promoted significant code re-use. The team did write 1 internal service by hand, but even that was a very small coding effort.</p>

<b>12. PROTOCOL/SPECIFICATION ISSUES</b>	<p>XML, WSDL, and SOAP are viable, stable protocols that demonstrated value in the proof-of-concept. UDDI proved to be of little value as a directory or repository in conjunction with Web Services. Performance was slow, the interface was confusing, and teams posted services in dissimilar fashion – GMU published three distinct services and others bundled all three services together in a single WSDL.</p> <p>As others will note, security and authentication were out of scope for the workgroup, but remain significant issues for the successful adoption of web services.</p>
<b>13. INTEROPERABILITY ISSUES</b>	<p>Dialogue or the lack thereof between the development teams had a significant bearing on the interoperability of the services. For example, several issues surfaced during testing related to consistently handling the data: name data, zip code, and common mappings among those.</p> <p>These and other discrepancies, like what codes represented which address types, were eventually resolved but only through a steady stream of communication.</p> <p>More time and a thorough development of the specifications could have solved these problems but would have somewhat contradicted the promise of 'loosely coupled' web services.</p> <p>It became clear during the project that three things need to be well publicized for a web service to work consistently and be valuable to the consumer: 1) a full description of the functionality; 2) expected valid inputs; and 3) well documented outputs.</p> <p>In general, the team discovered that the more detailed the specifications and the more communication between the teams, the greater the interoperability of the services. The services that worked best were tested and debugged when real-time instant messaging was used as a communication catalyst.</p>
<b>14. "BEST PRACTICE" COMMENTS</b>	<ul style="list-style-type: none"> <li>• Completion of specifications will result in greater interoperability between web services</li> <li>• Web Services should be well described and well publicized</li> <li>• Don't use UDDI – given the maturity of UDDI, a simple web site could have served the same functions</li> <li>• Communication, communication, communication</li> </ul>
<b>15. OTHER CONCERNS/ISSUES</b>	<ul style="list-style-type: none"> <li>• Team #6 had adequate time for development of the services, which demonstrates that web services can be produced in a relatively short timeframe.</li> <li>• Although adequate for the proof-of-concept, a real web services deployment would take much longer to thoroughly test among so many teams.</li> </ul>

<b>16. OTHER COMMENTS</b>	The project has been challenging for staff and the GMU/webMethods team strongly feels that web services will play a future in both internal and inter-agency application development and integration.
<b>17. COST/TIME ESTIMATE</b>	Meetings: 8 days Development: 14 days Training: 0 days Testing: 8 days

- 1) The *updatelog* service will record address changes into the address log database table
- 2) Services were recorded using webMethods Integration Server functions
- 3) webMethods Trading Networks software not used. Logic to call Agency services was coded

## Appendix L: UDDI Team Final Report – VIPNet & BEA

<p><b>1. PROJECT PLAN (SCHEDULE)</b></p>	<p>Attach high-level project plan with list of meetings, time and location giving percent complete.</p> <p>Install SOAP, XML, and WebLogic UDDI Server</p> <p>March 28, 2002 – Kick off meeting (VRS) (4.5 hours)  April 25, 2002 – Team selection meeting (VRS) (2.0 hours)  May 22, 2002 – Tim Bass meeting (VIPNet) (1.0 hours)  May 22, 2002 – Software AG meeting (VIPNet) (1.0 hours)  May 30, 2002 – Team presentations (VRS) (6.5 hours)  June 7, 2002 – WebLogic Server (VIPNet) (1.0 hours)  June 20, 2002 – Status meeting (VRS) (3.5 hours)  June 26, 2002 – Test data meeting (VIPNet) (1.0 hours)  June 27, 2002 – Test data meeting with Will Howery  (This one didn't take place) (VIPNet) (0.0 hours)  July 16, 2002 – Conference call with Gartner Group (1.0 hours)  July 18, 2002 – Status meeting (VRS) (3.0 hours)  August 19, 2002 – Final Team presentations and status  Meeting (DIT) (4.0 hours)</p> <p>.....</p> <p>UDDI up and running - June 7, 2002</p> <ul style="list-style-type: none"> <li>• Web Interface – 100 %</li> <li>• API Interface <ul style="list-style-type: none"> <li>○ Java - 100 %</li> <li>○ Perl - 80 % (Internal process)</li> </ul> </li> </ul>
<p><b>2. LIST OF TEAM NAME, MEMBERS, ROLES AND RESPONSIBILITIES</b></p>	<p><b>Virginia Information Providers Network (VIPNet)</b>  1111 East Main Street, Suite 901  Richmond, Virginia 23219  Phone: (804) 786-3814  Internet: <a href="http://www.vipnet.org">http://www.vipnet.org</a> or <a href="http://www.myvirginia.org">http://www.myvirginia.org</a></p> <p><b>Scott E. Fowler</b>, Director of Development (VIPNet) - Team Leader, Researching System Requirements, Security Issues, Logistical Support</p> <p><b>Will Howery</b>, Principle Systems Engineer, BEA Systems – Technical issues with WebLogic server, built Test Harness</p>

	<p><b>Billy Arnold</b>, Senior Systems Administrator (VIPNet) – Hardware setup, module installation, trouble shooting.</p> <p><b>Deanna Boehm</b>, Senior Application Developers/Project Mangers - Researching UDDI, Develop XML/SOAP Framework for UDDI Registry, Assist in problem resolution.</p> <p><b>Dave Neudeck</b>, Senior Application Developers/Project Mangers - Researching UDDI, Develop XML/SOAP Framework for UDDI Registry, Assist in problem resolution.</p>
<p><b>3. PROOF-OF-CONCEPT DESIGN</b></p>	<p>Attach general design of your application narrative, flow diagrams, etc. – whatever you think will adequately convey the team’s approach in terms of design.</p> <div data-bbox="548 821 1455 1499"> <p>The diagram illustrates the role of UDDI in a web services project. It features a central box labeled 'UDDI' at the top. Below it, on the left, is 'Agency A' (yellow oval) and on the right is 'Agency B' (yellow oval). Between them are 'Web Service' (purple box) and 'Application' (purple box). Arrows show the flow: Agency A registers service in UDDI directory (arrow to UDDI); Agency B searches UDDI to discover a desired service (arrow from UDDI); Agency B's application makes a request to Agency A's web service (arrow from Application to Web Service); Agency A's Web Service responds to Agency B's application's request (arrow from Web Service to Application); Agency B builds an application using discovered information in the UDDI (arrow from UDDI to Application); and Agency A Builds a Web Service (arrow from Agency A to Web Service).</p> </div> <p>Initially started working on building the UDDI component ourselves, quickly came to the conclusion we didn't have time. Installed Web Logics UDDI application.</p>
<p><b>4. BUSINESS/FUNCTIONAL REQUIREMENTS</b></p>	<p>Statement of the specific business/functional requirements that are driving the team’s application design.</p> <p>Private UDDI registry accessible by Virginia State Agencies and</p>



	<p>selected partners via web browser or API interface.</p> <p>Revised business/functional requirements: For this proof-of-concept, the Web Services Workgroup was going to need a place to store the information that each of the separate groups need. Under the Web Services definition there exists a component called UDDI, which will become the directory of information that needs to be shared. It was the UDDI teams responsibility to make the UDDI component available.</p>
<b>5. PROJECT EQUIPMENT DESCRIPTION</b>	<p>Full description of the hardware and software to be used, who is providing it and where it is located.</p> <p>Located &amp; Provided by: Virginia Information Providers Network Hardware: Sun Ultra II Operating System: Solaris 2.7 UDDI Server: BEA WebLogic Server 7.0 <b>Provided by: BEA</b> Modules: PERL 5.6.1 PERL Libraries 5.6.5 Apache 1.3.24 SOAP Lite 0.55 XML Parser 2.31 Java SDK 1.3.1</p>
<b>6. ACQUISITION AND INSTALLATION ACTIVITIES</b>	<p>Hardware for the Proof-of-Concept was already in house.</p> <ol style="list-style-type: none"> <li>1. Load operating system</li> <li>2. Load supporting software applications and modules</li> <li>3. Load WebLogic supporting modules</li> <li>4. Load WebLogic UDDI server</li> </ol>
<b>7. SERVER REQUIREMENTS</b>	<p>Describe changes to existing server or acquisition that will be needed before testing can begin. Describe the factors that influenced your decisions.</p> <p>In order to have BEA's WebLogic server running we needed to install a number of applications on the existing Sun Ultra II box. Before installing them we needed to get packages off of the Internet.</p> <p>UDDI Server: BEA WebLogic Server 7.0 Modules: PERL 5.6.1 PERL Libraries 5.6.5</p>

	<p>Apache 1.3.24 SOAP Lite 0.55 XML Parser 2.31 Java SDK 1.3.1</p>
<b>8. NETWORKING REQUIREMENTS</b>	<p>Describe changes to existing network or procurement that will be needed before testing can begin. Describe the factors that influenced your decisions.</p> <p><b>Established DNS entry <a href="http://www.uddi.state.va.us">www.uddi.state.va.us</a></b></p> <p><b>Firewall issues:</b></p> <p>Needed to open a translation port. Also opened port 7001 for admin control.</p>
<b>9. TRAINING REQUIREMENTS</b>	<p>Identify the technical and business training that has been or will be needed to prepare staff for proof-of-concept project.</p> <p>Overall there wasn't any training outside of individual research and consultations from vendors. All of the information that we used was found via the Internet, through books purchased, vendor consultations.</p> <ul style="list-style-type: none"> <li>• Purchased O'Reilly books on Web Services</li> <li>• Consultation from BEA on UDDI system</li> <li>• Consultation from Software AG</li> </ul> <p>Note: Consultations from vendors were good but we needed to have more of them. Given the timeline of this proof-of-concept, we found that juggling schedules was difficult.</p>
<b>10. TESTING PLAN</b>	<p>Attach a list showing all functions/processes to be tested. This should include descriptions of all testing scenarios and results.</p> <p>Testing was determined to be broken into two types, 1.) Static, and 2.) Dynamic. The reason for this split was based on time left to accomplish the proof-of-concept. The static testing would include accessing the WSDL of each group from the UDDI from the Web. The dynamic testing, which was left until the end, would allow an application to query the UDDI and retrieve WSDL documents and parse them correctly before the rest of the application would run.</p> <p>During the testing of the static and dynamic type, the functions for publishing, deleting, querying, and retrieving were tested.</p> <p><b>UDDI Test Plan</b></p>

These steps will be taken from both the web side and the API side.

1. Establish connection to UDDI.

Validate that logins work correctly.

2. For a given agency, publish each service (getAddress, changeAddress, queryAddress)

For the changeAddress. Publish with different address types e.g.  
Home, Mailing

3. Repeat step 1 with a different Agency name, login, and use different address types.

4. Search for changeAddress with address Types Home, should only return the correct service

5. Search for Search for change address with address Types Mailing, should only return the correct service

6. Delete a given Service and AddressType for a given agency.

7. Search for the deleted address and validate that it was deleted.

8. Logon as a different user and attempt to delete a service published by a different user and/or agency.

Functions focused on during static & dynamic type testing:

**Save functions**

- save\_business
- save\_services
- save\_binding
- save\_tModel

**Delete functions**

- delete\_business
- delete\_service

	<ul style="list-style-type: none"> <li>•delete_binding</li> <li>•delete_tModel</li> </ul> <p><b>Security functions</b></p> <ul style="list-style-type: none"> <li>•get_authToken</li> <li>•discard_authToken</li> </ul> <p><b>Find functions</b></p> <ul style="list-style-type: none"> <li>•find_business</li> <li>•find_service</li> <li>•find_binding</li> <li>•find_tModel</li> </ul> <p><b>Get Details functions</b></p> <ul style="list-style-type: none"> <li>•get_businessDetail</li> <li>•get_serviceDetail</li> <li>•get_bindingDetail</li> <li>•get_tModelDetail</li> </ul>
<p><b>11. DEVELOPMENT/TECHNICAL</b></p>	<p>High-level description of specific development/technical issues and/or difficulties encountered during coding.</p> <p>Technical issues:</p> <p><b>Installing previously listed modules.</b></p> <p>When installing modules there are usually other modules that need to be installed first before the initial one will install correctly. We spent a little bit of time figuring out which modules were missing, downloading them, installing them, and then trying the original module again.</p> <p><b>Keeping the UDDI Server up and running.</b></p> <p>There seems to be a problem with keeping the WebLogic server up and running. The reason for this problem is still unresolved.</p> <p><b>Licensing issues</b></p> <p>During the time of installation a service license was issued. That license expired and we had to get a new one.</p> <p><b>Memory overflow error message</b></p> <p>With the WebLogic server there seems to be a situation where the server, while booting, will cause a memory overflow error. This was documented and reported to BEA. The resolution for this problem</p>

	has not been found.
<b>12. PROTOCOL/SPECIFICATION ISSUES</b>	<p>Issues and/or difficulties specific to designing and coding for XML, SOAP, WSDL, and UDDI.</p> <p>Finding modules for SOAP and UDDI</p>
<b>13. INTEROPERABILITY ISSUES</b>	<p>Discuss issues and/or difficulties that involve service-to-service interoperability between the various team applications.</p> <p><b>Microsoft .Net and Sun JAXP toolkit problems.</b> The original version of the BEA WebLogic server seemed to have problems communicating with either the Microsoft .Net toolkit or the Sun JAXP toolkit. After Will Howery performed a number of test it was determined that the WebLogic server residing at VIPNet needed to be upgraded and have a patch installed. This was accomplished and the .Net and Sun problems were fixed. The interoperability concern here was because accessing the WebLogic server needed to be accomplished by standard mechanisms which Microsoft and Sun were using but they weren't being able to talk with the UDDI.</p>
<b>14. "BEST PRACTICE" COMMENTS</b>	<p>Technical or non-technical.</p> <p>When using another vendors software, be sure and have a couple of consultation meetings with them about the product you are going to use.</p>
<b>15. OTHER CONCERNS/ISSUES</b>	<p>Other team concerns/issues (technical, organizational, logistical or otherwise).</p> <p><b>Accessing the UDDI server.</b> There were a number of times that the UDDI server was down while different teams were testing. This brought up concerns on the availability of the UDDI server. Refer to section 11 for description of the problem.</p> <p><b>Publishing to the UDDI server.</b> This problem was related to what to publish and how to publish. Will Howery resolved this problem.</p> <p><b>Test Harness access.</b> The test harness access came at a bad time during the testing phase. There were a few people who were not able to use it.</p>

<b>16. OTHER COMMENTS</b>	<p>Comments on how the team project is going, successes, etc.</p> <p>This project has given us the ability to look into a practice that is maturing and determining whether or not it has a place in state government. We have gained a lot of experience but there is still a lot to learn.</p> <p>It is our opinion that UDDI will play a large role in the future of state government by giving access to information that used to be hard to get.</p>
<b>17. COST/TIME ESTIMATE</b>	<p>Track the time the full team spends for 1) Meetings, 2) Development, 3) Training, and 4) Testing.</p> <p>Meetings: 28 hours Development: 25 hours Training: 30 hours Testing: 18 hours Documentation: 30 hours</p> <p>Note: These hours don't include time spent by Will Howery.</p>

## Appendix M: Web Services Resources and Business Partners

The field of Web Services is one of the most rapidly evolving areas. Some dedicated web sites have been developed to provide information on Web Services, WSDL, UDDI and SOAP. The following provides a brief introduction to some of the resources that are available. The benefits achieved from Web Services mean that this field is in turmoil. The following web sites change daily, so visit them often.

**Web Services.ORG:** This web site is a central jumping-off point to everything related to Web Services. It includes News, Software, Events and Papers. Visit <http://www.webservices.org/>.

**UDDI.ORG:** This is the web site for the UDDI Registry and Repository. It provides full details of UDDI, with additional information on WSDL and SOAP. Visit <http://www.uddi.org/>.

**W3C.ORG:** The World Wide Web Consortium web site publishes Working Drafts, Recommendations and papers relating to all XML specifications. SOAP, UDDI and WSDL specifications and primer papers will be published here as they move through the W3C Specification process. For example, the “SOAP V1.2 Part 1: Messaging Framework” specification is at <http://www.w3.org/TR/2001/WD-soap12-part1-20011002/>, with “SOAP Version 1.2 Part 2: Adjuncts” at <http://www.w3.org/TR/2001/WD-soap12-part2-20011002/>.

**SOAPRPC.COM:** This is a web site that provides papers, news, software and resources for Web Services, SOAP, WSDL and UDDI. Visit <http://www.soaprpc.com/>.

**XML Cover Pages:** Robin Cover maintains a section of his XML Cover Pages web site dedicated to Web Services. He includes an abstract on each topic, with links to the topic detail and related information. Visit <http://www.oasis-open.org/cover/wsdl.html>.

**BEA WebLogic:** This section of the BEA site showcases BEA WebLogic Process Integrator products that may be downloaded for an evaluation. To download evaluation software, you must first register on the site. Visit [http://commerce.bea.com/downloads/weblogic\\_process\\_integrator.jsp](http://commerce.bea.com/downloads/weblogic_process_integrator.jsp)

**Microsoft on UDDI:** Microsoft has a UDDI web site that provides links to Microsoft UDDI resources, plus related resource links for UDDI, WSDL and SOAP. Visit <http://uddi.microsoft.com/developer/default.aspx>.

**webMethods:** This section of the webMethods web site provides white papers, free evaluation software and resources for Web Services. Visit <http://www.webmethods.com/content/1,1107,EnterpriseWebServices,FF.html>

**SilverStream:** This section of the SilverStream web site showcases the Novel eXtend (formally SilverStream eXtend) Web Services product made for full integration of web components. Visit [http://silverstream.com/Website/app/en\\_US/Extend](http://silverstream.com/Website/app/en_US/Extend)

**IBM on WSDL:** IBM offers many articles, resources, software and links from their DeveloperWorks web site. Visit <http://www-106.ibm.com/developerworks/web/library/w-wsdl.html?dwzone=web>. For example, a two part series of articles, titled: “Understanding WSDL in a UDDI Registry - Parts 1 and 2”

is available from <http://www-106.ibm.com/developerworks/webservices/library/ws-wsdl/?dwzone=webservices>.

To ensure awareness of any new resources that become available in this area, do a regular search of the Internet using the key words: “Web Services” SOAP WSDL UDDI.

There are many software business partners developing products and tools to support Web Services. A brief list, with links to relevant web sites, follows. A search of each business partner’s web site using the same key words as above will also yield valuable information.

**IBM Corporation:** IBM - with its other founding developers of Web Services, Microsoft and Ariba - jointly submitted initial Web Services specifications to the W3C for consideration in Sep 2000. IBM is developing extensive support for Web Services using WebSphere. Visit <http://www.ibm.com/xml/>. Also visit <http://www-106.ibm.com/developerworks/xml/library/ws-wscd.html>, where IBM has many articles on Web Services and XML technologies. A CD containing additional information and software can also be requested for this location.

**Microsoft Corporation:** Microsoft is using its “.NET” initiative (called ‘dot Net’) to transform the company - moving its software product functionality to the Internet. Web Services are integral to .NET, for real-time integration of services. For example, “Hailstorm” - partly released with Windows XP - offers some initial Web Services. Visit <http://www.microsoft.com/business/articles/net/netvision.asp> for an article discussing Microsoft’s vision, or visit <http://www.microsoft.com/net/> directly. Many articles are available, including online training and webcast seminars on all aspects of .NET. Microsoft also offer a free DVD containing the VisualStudio.net Beta 2, with 2 GB of .NET code samples.

**Software AG:** The Software AG EntireX Web Services Development Environment supports integration using many RPC technologies, including Web Services, Java, CORBA and COM. Search for “EntireX” from <http://www.softwareag.com/> or visit <http://www.softwareagusa.com/>. The Software AG Tamino XML Database also provides extensive XML development capabilities. Tamino is supplied within the Software AG XML Starter Kit, available for download, or on CD. Visit <http://www.softwareag.com/>.

**Hewlett-Packard:** HP is extending its e-Speak initiative to support Web Services and related languages. Visit <http://www.hp.com/> and search using the above key words. Many useful links are provided.

**SUN:** The Sun Open Net Environment (Sun.ONE) is being developed by Sun to support Web Services, as an answer to Microsoft .NET. Visit <http://www.sun.com/> and search using the above key words. Many relevant ONE links are available.

**Clear Case:** The CapeConnect Web Services Platform and CapeStudio Rapid Development Platform provide support for development and delivery of Web Services. Visit <http://www.j2ee-xml-ejb.com/>. We will look at product offerings, either released or in development, from many of the companies above in later issues of The Enterprise Newsletter.



## Appendix N: Web Services Address XML Schema

Address XML schema representation. The following structure, developed by the Web Services Working Committee, represents a sample generic address XML document as used in the Web Services Proof-of-Concept project.

```
<address>
  <addresstype>
    billing
  </addresstype>
  <addresseid>
    FEDIDXXXX
  </addresseid>
  <timestamp>
    xsd:dateTime
  </timestamp>
  <source>
    VABC
  </source>
  <address1>
    Appt 123
  </address1>
  <address2>
    12529 Swallow Creek rd
  </address2>
  <city>
    Richmond
  </city>
  <state>
    VA
  </state>
  <zip>
    23233
  </zip>
</address>
```

**addresstype (mandatory):** The type of address e.g: home, business, billing, garage, mailing

**addressid (mandatory):** identifier for an address typically the users social security number or a businesses tax id number 9 numeric digits in length.

**timestamp xsd:dateTime:** time that the address was last updated. Used in the log and query results

**source:** source URL of address change. Used in the log and query results.

**address1 (mandatory):** first line of an address. alphanumeric

**address2:** second line of an address. alphanumeric

**city (mandatory):** city name. Alphanumeric

**state (mandatory):** 2 character state abbreviation

**zip:** 5 digit zip code, optionally could contain 9 digit zip code with a hyphen. NNNNN-NNNN

When an address xml document is received by an agency that document will be handled in the most appropriate mechanism for that agency. The agency will also keep an operation log of the document. This log will fulfill two purposes. One it will provide the bases for sending back address changes to a queryAddress request and two, it will provide a logging instance to test and validate that the service received all addresses sent to it. After the address change WebServices have been deployed a test can be run with all changes going to a log. This log can then be analyzed to validate that each agency had in fact propagated address changes and received any address changes propagated.

```
<addresslog>
  <address>
    <addresstype>
      billing
    </addresstype>
    <addresseid>
      FEDIDXXXX
    </addresseid>
    <timestamp>
      xsd:dateTime
    </timestamp>
    <source>
      VABC
    </source>
    <address1>
      Appt 123
    </address1>
    <address2>
      12529 Swallow Creek rd
    </address2>
    <city>
      Richmond
    </city >
    <state>
      VA
    </state >
    <zip>
      23233
    </zip>
  </address>
  <address >
    .
    .
    .
  </address>
</addresslog>
```

The addresslog schema example will contain multiple address entries. Reusing the address structure will reduce the effort in managing the log entries. The only difference in the address log entry from the address schema will be the mandatory addition of a source and a time stamp. The source will identify the url which sent this message to the receiving agency. The time stamp will be in the Year/Month/Day Hour:Minute:Second format as yyyy/mm/dd hh:mm:ss. For purposes of the testing we can use Virginia local time. If the system were to be expanded we would want to record GMT or include a time zone identifier.

WSDL document:

The following WSDL implementation will represent the interfaces discussed in the workshop meeting. As follows:

```
public interface AddressChangeWebService {
public AddressData getAddress(String addressID, String addressType) throws
    InvalidAddressID;
public boolean changeAddress(AddressData aAddressData, String OriginatorURL);
public AddressLog queryAddress(Date startDate, Date endDate, String addressType, String
    OriginatorURL);
```

The WSDL document defines the AddressData XML structure. The AddressLog coming from the queryAddress still needs some work to not be more portable.

```
<?xml version="1.0" encoding="utf-8" ?>
<definitions xmlns:s="http://www.w3.org/2001/XMLSchema"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:tns="http://whowery:7001" targetNamespace="http://whowery:7001"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>
    <xsd:schema attributeFormDefault="qualified" elementFormDefault="qualified"
      targetNamespace="java:language_builtins.util"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:stns="java:language_builtins.util">
      <xsd:complexType name="Vector">
        <xsd:complexContent>
          <xsd:restriction base="soapenc:Array"
            xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
            <xsd:attribute ref="soapenc:arrayType"
              wsdl:arrayType="xsd:anyType[]"
              xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
          </xsd:restriction>
        </xsd:complexContent>
      </xsd:complexType>
```

```
</xsd:schema >
= <xsd:schema attributeFormDefault="qualified" elementFormDefault="qualified"
  targetNamespace="java:com.bea.sesouth.webservices"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:stns="java:com.bea.sesouth.webservices">
  <xsd:import namespace="java:language_builtins.util" />
= <xsd:complexType name="AddressLog">
  = <xsd:sequence>
    <xsd:element name="addressVector" maxOccurs="1"
      type="tp:Vector" minOccurs="1" nillable="true"
      xmlns:tp="java:language_builtins.util" />
    <xsd:element name="originator" maxOccurs="1"
      type="xsd:string" minOccurs="1" nillable="true" />
  </xsd:sequence>
</xsd:complexType>
= <xsd:complexType name="AddressData">
  = <xsd:sequence>
    <xsd:element name="originator" maxOccurs="1"
      type="xsd:string" minOccurs="1" nillable="true" />
    <xsd:element name="address2" maxOccurs="1"
      type="xsd:string" minOccurs="1" nillable="true" />
    <xsd:element name="state" maxOccurs="1" type="xsd:string"
      minOccurs="1" nillable="true" />
    <xsd:element name="name" maxOccurs="1" type="xsd:string"
      minOccurs="1" nillable="true" />
    <xsd:element name="zip" maxOccurs="1" type="xsd:string"
      minOccurs="1" nillable="true" />
    <xsd:element name="address1" maxOccurs="1"
      type="xsd:string" minOccurs="1" nillable="true" />
    <xsd:element name="addressID" maxOccurs="1"
      type="xsd:string" minOccurs="1" nillable="true" />
    <xsd:element name="city" maxOccurs="1" type="xsd:string"
      minOccurs="1" nillable="true" />
    <xsd:element name="updateDate" maxOccurs="1"
      type="xsd:dateTime" minOccurs="1" nillable="true" />
    <xsd:element name="addressType" maxOccurs="1"
      type="xsd:string" minOccurs="1" nillable="true" />
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>
</types>
= <message name="changeAddress">
  <part name="addressData"
    xmlns:partns="java:com.bea.sesouth.webservices"
    type="partns:AddressData" />
  <part name="string" xmlns:partns="http://www.w3.org/2001/XMLSchema"
    type="partns:string" />
</message>
= <message name="changeAddressResponse">
  <part name="result" xmlns:partns="http://www.w3.org/2001/XMLSchema"
    type="partns:boolean" />
```

```
</message>
= <message name="getAddress">
  <part name="string" xmlns:partns="http://www.w3.org/2001/XMLSchema"
    type="partns:string" />
  <part name="string0"
    xmlns:partns="http://www.w3.org/2001/XMLSchema"
    type="partns:string" />
</message>
= <message name="getAddressResponse">
  <part name="result" xmlns:partns="java:com.bea.sesouth.webservices"
    type="partns:AddressData" />
</message>
= <message name="queryAddress">
  <part name="date" xmlns:partns="http://www.w3.org/2001/XMLSchema"
    type="partns:dateTime" />
  <part name="date0" xmlns:partns="http://www.w3.org/2001/XMLSchema"
    type="partns:dateTime" />
  <part name="string" xmlns:partns="http://www.w3.org/2001/XMLSchema"
    type="partns:string" />
  <part name="string0"
    xmlns:partns="http://www.w3.org/2001/XMLSchema"
    type="partns:string" />
</message>
= <message name="queryAddressResponse">
  <part name="result" xmlns:partns="java:com.bea.sesouth.webservices"
    type="partns:AddressLog" />
</message>
= <portType name="AddressChangeBeanPortType">
  = <operation name="changeAddress">
    <input message="tns:changeAddress" />
    <output message="tns:changeAddressResponse" />
  </operation>
  = <operation name="getAddress">
    <input message="tns:getAddress" />
    <output message="tns:getAddressResponse" />
  </operation>
  = <operation name="queryAddress">
    <input message="tns:queryAddress" />
    <output message="tns:queryAddressResponse" />
  </operation>
</portType>
= <binding name="AddressChangeBeanSoapBinding"
  type="tns:AddressChangeBeanPortType">
  <soap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http" />
  = <operation name="queryAddress">
    <soap:operation soapAction="" />
  = <input>
```

```
<soap:body use="encoded" namespace="http://whowery:7001 "
  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
/>
</input>
= <output>
  <soap:body use="encoded" namespace="http://whowery:7001 "
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  />
</output>
</operation>
= <operation name="changeAddress">
  <soap:operation soapAction="" />
  = <input>
    <soap:body use="encoded" namespace="http://whowery:7001 "
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    />
    </input>
  = <output>
    <soap:body use="encoded" namespace="http://whowery:7001 "
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    />
    </output>
  </operation>
= <operation name="getAddress">
  <soap:operation soapAction="" />
  = <input>
    <soap:body use="encoded" namespace="http://whowery:7001 "
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    />
    </input>
  = <output>
    <soap:body use="encoded" namespace="http://whowery:7001 "
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    />
    </output>
  </operation>
</binding>
= <service name="AddressChangeBean">
  <documentation>todo: add your documentation here </documentation>
  = <port name="AddressChangeBeanPort "
    binding="tns:AddressChangeBeanSoapBinding">
    <soap:address
      location="http://localhost:7001/state.va.ws.addresschange/AddressChangeBean?WSDL " />
    </port>
  </service>
</definitions>
```

## END NOTES

---

<sup>i</sup> Special Supplement to CIO The Magazine for Information Executives, “The Promises of Web Services” April 2002, *CIO The Magazine for Information Executives*, CXO Media, Inc, <http://www.cio.com/sponsors/041502ws/new.html>

<sup>ii</sup> XML is a project of the [World Wide Web Consortium \(W3C\)](#), and the development of the specification is being supervised by their XML Working Group. A Special Interest Group of co-opted contributors and experts from various fields contributed comments and reviews by email. XML is a public format: it is not a proprietary development of any company. [The v1.0 specification](#) was accepted by the W3C as Recommendation on Feb 10, 1998.

<sup>iii</sup> XML is a project of the [World Wide Web Consortium \(W3C\)](#), and the development of the specification is being supervised by their XML Working Group. A Special Interest Group of co-opted contributors and experts from various fields contributed comments and reviews by email. XML is a public format: it is not a proprietary development of any company. [The v1.0 specification](#) was accepted by the W3C as Recommendation on Feb 10, 1998.

<sup>iv</sup> XML is a project of the [World Wide Web Consortium \(W3C\)](#), and the development of the specification is being supervised by their XML Working Group. A Special Interest Group of co-opted contributors and experts from various fields contributed comments and reviews by email. XML is a public format: it is not a proprietary development of any company. [The v1.0 specification](#) was accepted by the W3C as Recommendation on Feb 10, 1998.

<sup>v</sup> Whit Andrews, “Web Services In Action” August 12, 2002, GartnerGroup, Inc. <http://www4.gartner.com>

<sup>vi</sup> Whit Andrews, “Web Services In Action” August 12, 2002, GartnerGroup, Inc. <http://www4.gartner.com>

<sup>vii</sup> Elena Larsen, Lee Rainer, “The Rise of the E-Citizen, How people use government agencies’ Web sites” April 3, 2002 [http://www.pewinternet.org/reports/pdfs/PIP\\_Govt\\_Website\\_Rpt.pdf](http://www.pewinternet.org/reports/pdfs/PIP_Govt_Website_Rpt.pdf)

<sup>viii</sup> Sheller Brown, “What’s New in the Office of Intergovernmental Solutions?” June 26, 2002 [http://www.gsa.gov/Portal/content/news\\_content.jsp?contentOID=114888&contentType=1003&PMGZ=1](http://www.gsa.gov/Portal/content/news_content.jsp?contentOID=114888&contentType=1003&PMGZ=1)

<sup>ix</sup> Governor Mark E. Warner, Press Release, “Virginia Climbs to Six in National Technology Rankings” July 1, 2002 [http://www.governor.state.va.us/Press\\_Policy/Releases/July2002/0702.htm](http://www.governor.state.va.us/Press_Policy/Releases/July2002/0702.htm)

<sup>x</sup> Governor Mark E. Warner, Press Release, “Virginia Ranks #1 in Digital State Survey” June 7, 2002 [http://www.governor.state.va.us/Press\\_Policy/Releases/June2002/Jun0702.htm](http://www.governor.state.va.us/Press_Policy/Releases/June2002/Jun0702.htm)